# Order Assignment Heuristic in a Build to Order Environment

**by**

**Aviv Cohen**
B.S. Computer Science, Statistics and Operations Research
Tel Aviv University, 1997

Submitted to and the Sloan School of Management
and the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of

**Master of Science in Electrical Engineering and Computer Science**

and

**Masters of Business Administration**

at the
Massachusetts Institute of Technology
June 2001





Signature of Author _____
Sloan School of Management
Department of Electrical Engineering and Computer Science
May 11, 2001

Certified by _____
Stephen C. Graves
Abraham Siegel Professor of Management Science
Thesis Supervisor

Certified by _____
Richard C. Larson
Professor of Electrical Engineering
Thesis Supervisor

Approved by _____
Margaret Andrews
Director of Master's Program
Sloan School of Management

Approved by _____
Arthur C. Smith
Chairman, Committee on Graduate Studies
Department of Electrical Engineering and Computer Science

# Order Assignment Heuristic in a Build to Order Environment

by

**Aviv Cohen**

Submitted to the Department of Electrical Engineering and Computer Science and the Sloan School of Management on May 11, 2001 in Partial Fulfillment of the Requirements for the Degrees of Master of Science in Electrical Engineering and Computer Science and Masters of Business Administration

## Abstract

Dell Computer Corporation, renowned for its "direct model" and build to order strategy, is using a new software package for its supply chain management and order fulfillment processes. This leap forward fortifies Dell's position on the leading edge of the manufacturing industry. One of the sub-processes in this new supply chain software package is the assignment of incoming orders to one of multiple assembly lines in a production facility. The goal of this thesis is to explore this problem and to develop an effective and simple to implement solution.

The paper describes a heuristic solution approach as well as other approaches. In specific, it examines several offline Integer Programming formulations and variations of an online heuristic. Since we are dealing with a multiobjective optimization problem, an improvement on one dimension typically means a compromise on another dimension. The literature discusses three basic approaches to such problems. The first is assigning a cardinal measure to each objective and determining a weighted average function (or any other function, for that matter) of all objectives. The goal is then to optimize the value of this function. The second approach is to assign a lexicographical order to the objectives and optimize one after the other. The third approach is to attempt to achieve certain goals for each objective. The recommended solution integrates certain elements from each approach in a manner that is consistent with the decision makers business logic.

Thesis Supervisors:
Stephen C. Graves, Abraham Siegel Professor of Management Science
Richard C. Larson, Professor of Electrical Engineering

## Acknowledgements

I would like to extend my gratitude to Dell Computer Corporation for providing the forum for this thesis research. Special thanks go to Kevin Jones whose support, insightfulness and patience enabled my progress during this research. Thanks also go to Steve Cook and Kris Vorm who assisted me during my time at Dell.

Additionally, I would like to thank my two MIT advisors, Steve Graves of Sloan, who endured the Texas heat twice when visiting, and Richard Larson of EECS.

The work presented in this thesis was performed with support from the Leaders For Manufacturing Program at MIT.

# Table of Contents

# Table of Figures



# Table of Tables

# 1 Chapter One: Introduction and Background

## 1.1 Thesis Overview

The first two chapters of this thesis provide the background to the order assignment problem at Dell Computer Corporation. They explain both the general strategy and history of the company and describe the circumstances before and after the implementation of the new software management system.

Chapters three and four specify the particulars of the assignment problem and the explored approaches and solutions. They investigate the tradeoffs between different approaches as well as various solutions that address different concerns.

The last two chapters analyze the proposed solution as well as its implications for the company. Chapter five focuses on analysis by simulation of the proposed heuristic. Chapter six provides a conclusion and discusses directions for further research.

## 1.2 Company Background

Dell Computer Corporation, headquartered in Austin, Texas, is the world's leading direct computer systems company. Dell Computer has the highest market share in the US personal computer (PC) market and the number two position worldwide. The company has approximately 37,000 employees worldwide and had revenues of $25.2 billion in fiscal year 2000. Dell Computer's phenomenal growth, especially during the mid 1990s, has long been legendary in the industry.

Michael Dell, the computer industry's longest tenured chief executive officer, founded the company in 1984. His strategy from the very beginning was to sell directly to end-users. By eliminating the retail markup, Dell's new company was able to sell PCs at about 40 percent below the competition's price. By 1985, the company had 40 employees and by 1986, sales had reached $33 million. In 1988, Dell Computer added a sales force to serve large customers, began selling

to government agencies and became a public company, raising $34.2 million in its first offering. Sales to large customers quickly became the dominant part of company's business. By 1990, Dell Computer had sales of $388 million and a market share of 3 percent in the US.

Fearing that direct sales would not grow fast enough, the company began distributing its products through Soft Warehouse Superstores (now CompUSA), Staples, Wal-Mart and other retail chains. Dell also sold PCs through Xerox in 19 Latin American countries. However, the company realized it had made a mistake when it learned how thin its margins were selling through these channels. It discontinued selling to retailers and other intermediaries in 1994 and refocused on direct sales again. Dell started to pursue the consumer market aggressively only when the company's Internet site became a valuable distribution channel around 1996.

Dell currently sells PC products, services and peripherals. Products include desktop and notebook computers for corporate customers and home users. In addition, Dell is presently gaining increasing market share in the workstation, storage and server markets. The company also offers a variety of services such as factory installation of proprietary hardware and software, leasing and system installation and user support. Lastly, Dell sells software and peripheral products to complement its systems offering.

As of 2000, about 50 percent of Dell's sales are Web-enabled, and about 76 percent of Dell's order-status transactions occur online. Approximately 65 percent of Dell's revenue are generated through medium and large business and institutional customers. The company's computers are manufactured at facilities in Austin, Texas; Nashville, Tennessee; Eldorado do Sul, Brazil; Limerick, Ireland; Penang, Malaysia; and Xiamen, China.

### *1.3 Direct at Dell*

Dell Computer Corporation was founded on a simple concept: direct sales. This concept had two distinct advantages. First, it enabled the company to eliminate intermediary resellers, thus offering better prices and owning the relationship with the customers. Second, build to order

generally reduced costs and risks associated with keeping large inventories of components and finished goods. This strategy allowed the company to enjoy significant cost and profit advantages over the competition.

Dell strives to achieve "virtual integration" – integration of the company, its suppliers and its customers in real time. Successful virtual integration makes it possible for all three to appear as part of the same organizational structure. Accordingly, Dell's strategy revolves around several core elements:

- Build to Order Manufacturing and Mass Customization

Dell builds all its computers, servers and workstation to order. Customers can order custom built configurations of chassis type, microprocessor, memory, and so forth based on their needs and desires. Orders are typically directed to the nearest factory for immediate build. Since 1997, Dell shifted to "cell manufacturing", whereby a team of workers assemble an entire computer according to customer specifications. Assembled computers are tested, loaded with software and shipped to their destination. The sell-direct strategy means that Dell has no in-house inventory of finished goods.

- Partnership with Suppliers

Dell chose to partner with reputable suppliers of parts and components rather than to integrate backward. The company believes that long-term partnerships with such suppliers yield several advantages. First, using high quality components enhances the quality and performance of Dell's products. The brand of the components is more important to some buyers than the brand of the system itself. Dell's strategy is to stay with a few leading vendors as long as they maintain their leadership. Second, Dell commits to purchasing a certain percentage of its needs from each of these vendors. Thus, it has higher precedence in getting the volume it needs even when there's a temporary shortage in the market. Third, engineers from the suppliers are assigned to Dell's

product design teams – enabling Dell to have more successful product launches. Fourth, this deep-rooted partnership enables just-in-time delivery of supplies to Dell's assembly plants. Some of the suppliers set up "distribution hubs" within miles of Dell's plants and can deliver hourly. Dell openly shares its production schedules, forecasts and new product introduction plans with its vendors.

- Just In Time Components Inventories

Dell's just in time inventory yields significant cost benefits but, at least as importantly, it shortens the time it takes Dell to introduce new generation of computers to the market. New advances in system components often make items in inventory obsolete within months. Moreover, component prices have been falling as much as 50 percent annually. Collaboration with suppliers allows Dell to operate with only a few days, or even a few hours of inventory, of some components. In some instances, Dell operates with no inventories at all, when it merges shipments of computers from the factories with deliveries of other components, such as monitors, directly from the suppliers.

- Direct Sales

Dell enjoys a direct first-hand relationship with its customers. As a result, it benefits from having valuable information about its customers' preferences and needs, as well as immediate feedback about any quality issues. The company believes that its ability to respond quickly gives it an important edge over its competitors. It sees direct sales as an entirely customer driven approach that permits swift transitions to new generations of computer models and components.

- Information Sharing

As mentioned, Dell puts a great emphasis on information sharing with both suppliers and customers. By using the latest information technology, Dell has always made efforts to blur the boundaries in the traditional supplier-manufacturer-customer value chain, and achieve "virtual

integration". For example, new software makes it easy for Dell to communicate inventory levels and replenishment needs to vendors hourly.

On the customer front, there are several initiatives. A number of Dell's corporate accounts are large enough to justify dedicated on-site teams of Dell employees. Customers usually welcome such teams, preferring to focus their time and energy on their core business. In addition to using its sales and support mechanisms, Dell has set up a number of forums to stimulate the flow of information with customers.

Dell also develops customized intranet sites for its largest global customers; these sites give immediate on-line access to purchasing and technical information about the specific configurations that their company had purchased from Dell or that were currently authorized for purchase. The sites contain all of the elements of Dell's relationship with the customer - detailed product descriptions, software loaded on each of the products the customer purchased, service and warranty records, pricing, and the available technical support. These features eliminate paper invoices, cut ordering time, and reduce the internal labor needed to staff corporate purchasing functions.

## *1.4 Traditional Process at Dell*

Assembly of all computer products at Dell typically follows the same general process. This process commences with the receipt of supply shipments at the factory dock doors and ends when the finished goods are shipped to the customers. Since most materials are not warehoused, assembly lines are replenished directly from the dock doors, where the physical receipt transaction is completed.

The order is initiated when a "traveler", a form that contains the specifications of a particular system, is "pulled" (see Figure 1-1). An order, by Dell terms, consists of up to 50 identical systems (due to historical reasons). Material availability is immediately verified for the

pulled order. If material is available, the kitting process begins. All internal parts and components are picked from pick-to-light (PTL) racks and placed into a tote.

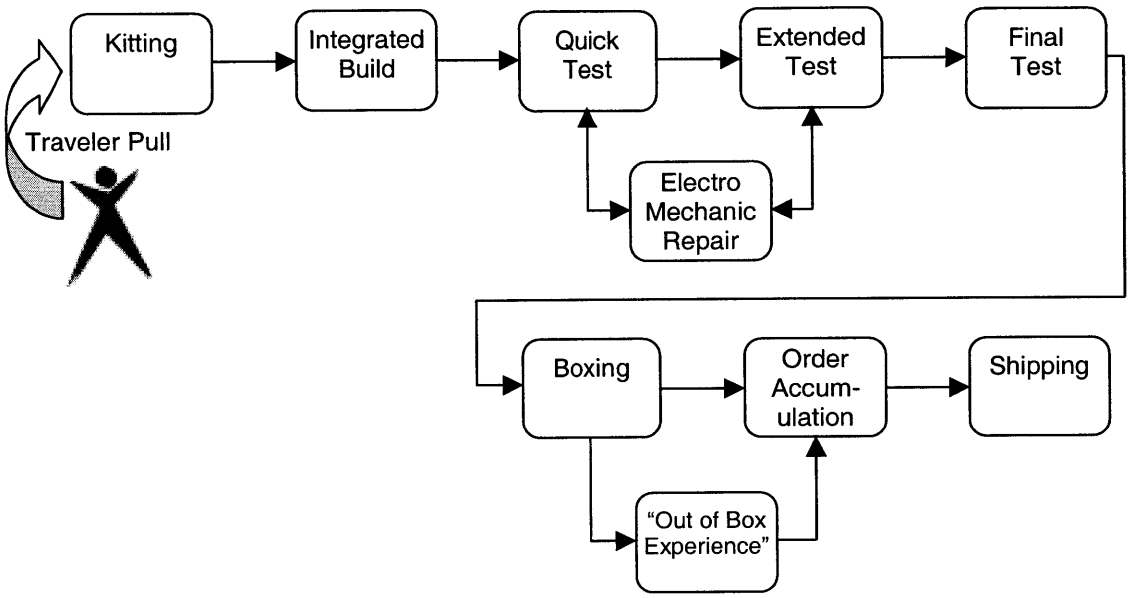


**Figure 1-1: Traditional Process Flow**

The completed tote is sent to an integrated build cell. A team of two cell operators completely assembles the system and then performs basic quality tests (Quick Test). The system is powered up to ensure it is functional. Since the assembly operator and the Quick Test operator are all in the same cell they can provide immediate feedback. If electromechanical problems are observed during Quick Test, an electromechanical repair (EMR) specialist will attend to the system. Next, the systems are placed in burn racks for thorough diagnostic testing and software download ("Extended Test").

The tested and "burned" systems are then subject to a final test before they go into boxing. The systems are verified complete and components are functionally checked. If there are system integration requirements, such as factory installation of additional proprietary or commercial software and hardware (offered as a premium service), they are performed after the

final test. The systems are then subject to a concluding external inspection. Completed systems are placed in boxes and sent down another PTL line to be packed with peripherals such as mouse, keyboard, power cords, documentation, etc. The box is then either placed directly on a truck or put in an accumulation area until all the systems for that order are completed. Sample systems are taken out of their boxes and examined thoroughly, imitating the customer's experience when she receives the system ("Out of Box Experience").

Dell factories have a variety of configurations depending on the product family and geography. The factories that produce very high volume products such as the desktop products ("Optiplex" and "Dimension") typically have around 8 to 12 kitting areas feeding 4 to 6 assembly cell clusters respectively. Each cluster has 4 to 6 assembly cells. In most facilities two kitting lines feed an assembly cluster, which feeds a dedicated boxing line. In some facilities the relationship is not one to one.

# 2 Chapter Two: The New Demand Fulfillment System

## *2.1 Goals*

In order to achieve better operational performance, Dell Computer decided to implement a new supply chain management and demand fulfillment system. By using this system, henceforth the "new system", the company tried to meet design goals that would improve significantly the company's capabilities. The prevailing vision included even tighter, more just-in-time integration with the suppliers, enabling more efficient manufacturing processes, as well as improved delivery target performance to enhance the customer experience.

- **Reduced Inventory** - The first and most dominant objective was to continue to reduce inventory levels. In some cases, the goal was even to eliminate the need for an in-factory warehouse. Such reduction in inventory would clearly allow Dell to use less floor space, reduce inventory holding costs and decrease headcount. At the same time the company wanted to have current knowledge of the inventory levels in its supplier hubs, in order to know which orders could be built and in order to indicate to the supplier when more supplies are needed.
- **Reduced Material Handling** – Internal material handling and parts movement between the assembly lines introduced unnecessary complications into the manufacturing processes. Since replenishment had been based on not necessarily accurate forecasted quantities, materials often had to be shifted from one kitting line to another. Often parts that had been delivered to the factories were not required by actual orders waiting to be built.
- **Controlled Prioritized Schedule Sequence** – Dell saw it as necessary to replace the "surf to download" sequencing process. This process, which was almost entirely manual, allowed the supervisor of an assembly line to decide on the spot which orders, out of the

ones available to build, her assembly line would build. In addition, the assignment process of orders to assembly lines was very simplistic, purely based on configuration type. This led to unbalanced workloads on the different assembly lines and avoidable downtime. Although, as mentioned, the supervisors did employ business logic to their decision making process, there were still incidents of "forgotten" orders. These orders were typically of smaller quantities and less common configurations. Incidents where orders were completed more than a month after they had originally been available to build were not unheard of. There were no processes intact to enforce a sequence that mirrored the company's business priorities. Frequently, materials turned out not to be available for orders that had already been initiated by the supervisors. Lastly, a controlled schedule would reduce cycle time variability.

## *2.2 New Systems and Processes*

### 2.2.1 New System

The new system enables a transformation that meets the objectives that had been set forth previously. This system makes possible to employ a "pull to order" approach. Thus, only materials actually needed for specific scheduled orders will be brought into the facility. This tactic prevents unnecessary inventory from accumulating within the facility walls. Furthermore, replenishment is done on a line-by-line basis. Trucks carry various parts from the hubs and are destined to replenish one specific line. This policy assists in eliminating movements of material within the lines, as well as headcount of the personnel involved. Such a complex operation is supported by several information technology components.

The system also allows for prioritized and controlled schedules. Only orders whose materials are available are scheduled for production. The schedule is based on firm business rules

that meet the general priorities of the company in terms of due dates and additional customer needs.

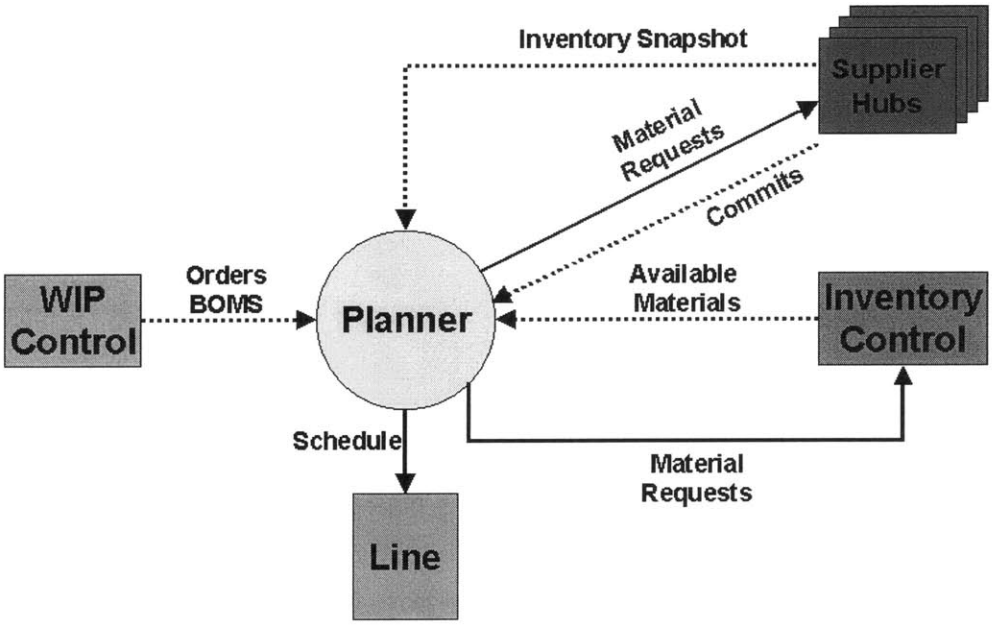


**Figure 2-1: Architecture of the New System**

The new production scheduling and material replenishment processes rely on a new logical architecture. This architecture consists of several basic elements, as shown in Figure 2-1.

1. **Work-in-Process Tracking** – This system keeps track of the stages each order is in. In particular, this system contains the information about orders that become available to build, pursuant to financial and other approvals by the sales departments.
2. **Inventory Control** – This system governs the movement and transaction of inventory. This system maintains all information relating to parts including their receipt from suppliers, their storage location, their movement from one stockroom to another, and so forth.
3. **Hub Collaboration** – This system enables the factory to view current inventory levels present at the supplier hubs.

4. **Planner** – This system is the heart of the system. The Planner uses inputs from the previously mentioned systems and creates feasible plans to balance supplies and demands. It considers material and capacity constraints concurrently and creates a feasible production schedule. The Planner outputs a production schedule to the assembly lines and the replenishment requirements to the stockrooms and the supplier hubs.

The new system is a "decision support system" rather than a "decision making system". A decision making system would be expected to determine and establish the schedule on its own. However, this system is designed to be a decision support tool, which enables the human decision maker to evaluate several alternatives as she sees fit. Dell is different than other companies in its high frequency of planning cycles. While other companies in a variety of industries typically have a planning cycle once a day, a few days or even a week, Dell's planning cycle typically commences every couple of hours. This is due in large part to the company's build to order production strategy. Therefore, Dell effectively uses this system for decision making and not decision support, due to the lack of time to evaluate different scenarios. By being on the leading edge of planning capabilities, Dell may be sacrifice some schedule performance in order to achieve a high frequency of planning cycles.

The new demand fulfillment process represents a fundamental shift in Dell factory replenishment. The schedule is produced by the Planner using the factory demand, actual order data, supply, available inventory on hand at the supplier hubs and static data, bills of material, routings, resources and more.

### 2.2.2 Demand Fulfillment Process

The demand fulfillment process follows the following steps (a general overview is shown in Figure 2-2):

1. The Planner receives the orders - Available to build orders and their bills of material (BOMs) are loaded into the planner.

2. Inventory snapshots are fed into the Planner – Each supplier hub creates a snapshot of its available on hand inventory position on an ongoing basis. Inventory Control data is also fed into the Planner.

3. The Planner generates the build plan – The Planner creates a model and derives the build plan.

4. The Planner assigns material request delivery times to the factory – In order to have a staggered delivery schedule (detailed later) the Planner must assign delivery times to the factory to each order and group the requirements according to their destination kitting line.

5. Requests sent to supplier hubs – A request is sent to each supplier hub that holds materials required by the build plan.

6. Supplier hubs respond to requests – The hubs perform a comparison of their inventory position relative to the material requirements received. Within minutes, the hubs commit to a quantity and time of delivery.

7. Production Control intervention – Production Control looks for any outstanding issues with the requested parts. Problems could include commits that were not received, under committed quantities, delayed delivery time and delayed receipts.

8. Supplier hubs pick parts – The hubs pick parts and assemble pallets according to the commitments. Orders are expected at their requested delivery locations within 90 minutes of the hub receiving the material requirements.

9. The Planner generates the detailed, order-by-order schedule.

10. Delivery Confirmation – all received material deliveries are confirmed to avoid reordering of parts during the next planning cycle.

Figure 2-2 provides an overview of the scheduling cycle. Note that the trucks carrying supplies (A, B an C) are destined to replenish the kitting PTLs as well as the boxing PTLs.

The (human) scheduler has to maintain the Planner model data. This data includes a record of all the production resources available and the different routings between stations possible in the factory and additional flags.

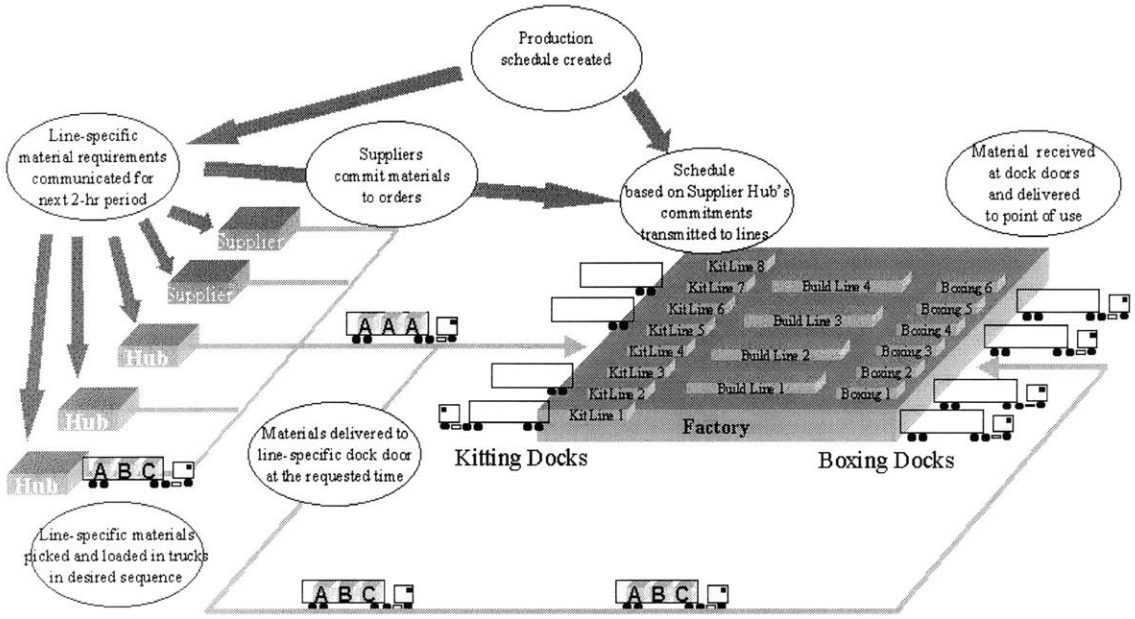


**Figure 2-2: Overview of Scheduling Cycle**

### 2.2.3 Staggered Delivery Schedule

The Staggered Delivery Schedule aligns trade deliveries with material requests by line. The program buckets parts into delivery times, thus it is essential to define the time frame and parameters to ship parts for each delivery on each dock door and to each kitting line. Bucketing is based on the supplier hub, dock door and resource. Bucketing times are stored in the database and can vary. Staggered delivery to point of use reduces material movement, as mentioned, and decreases headcount of personnel involved with the unpacking of the shipments. See Figure 2-3 for example.

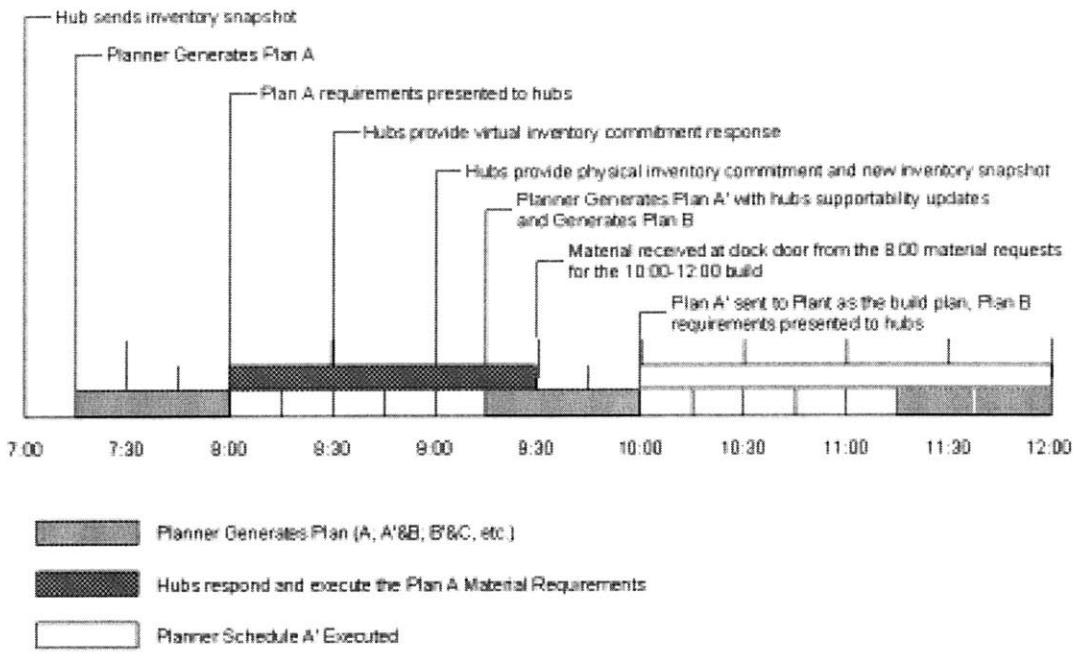


**Figure 2-3: Staggered Schedule of Two Schedules**

### 2.2.4 Schedule Creation

1. Ongoing data is fed to the Planner's database: new orders that become available to build from the Order Tracking system, near real time snapshots of supplier hub's inventory and in factory inventory from Inventory Control.

2. As new orders become available to build they are assigned a routing. This routing determines the sequence of production resources the order has to pass through during its assembly process. This assignment algorithm precedes the actual scheduling algorithm. The work described in this paper concentrates on this algorithm.

3. The first step inside the Planner is to create a sequence of all orders based on their priorities. The sequence is prioritized based on three criteria. The most important criterion is the business priority – determined by business decision makers, then earliest due ship date, and lastly smaller quantities. Additional business rules can be applied: special order types, geographical destination, etc.

4. The Planner now looks for materials to assign to orders starting with the first order in the sequence. It assigns the parts that are available the earliest. The Planner first looks for parts in the kitting area of that order's assigned line. If there are not enough parts for the order there it will then look at the excess stockrooms. If parts are still not found, the Planner will look for them in the supplier hubs inventories. In the event that parts are still not found, the Planner will search for parts in kitting areas of other lines. If the parts are not found anywhere, the Planner will "find" them in an "infinite supply pool" as a last resort. This is done in order to make all orders "plannable".

5. Once materials have been assigned to each order, the Planner creates a plan that is not constrained by capacity. This plan puts the orders in sequence without taking the limited production capacity into account. Several points in time are calculated:

a. Earliest Possible Start Time (EPST): The earliest time a manufacturing order can begin processing at a specific operation. EPST for an operation is equal to the sum of two values: the maximum of the present time and the earliest time when all material is available – and the minimum cycle time required to process material at all prior operations. EPST does not incorporate the impact of resource constraints on the earliest start time. EPSTs are assigned during the capacity-unconstrained plan.

b. Latest Possible Start Time (LPST): The latest time a manufacturing order can begin processing at a specific operation and still plan to complete the order prior to its factory due date. LPST for an operation is equal to: the factory due date for the manufacturing order minus the minimum cycle time required to process material for the operation and all subsequent operations. LPST does not incorporate the impact of resource constraints on the latest start time. LPSTs are also assigned during the capacity-unconstrained plan.

c. Planned Start Time (PST): The time when a specific task or operation for a manufacturing order is expected to start. When not associated with a particular operation, PST may also be used to define the planned start time for the first operation of a manufacturing order.

At this stage, when capacity is still not considered, the PST is equal to the LPST (Figure 2-4) unless LPST is earlier than the EPST. Otherwise PST is equal to the EPST (Figure 2-5). Furthermore, Dell requires that all operations on a specific system occur one after the other so as not to leave work in progress between operations. The bars in Figures 2-4 and 2-5 show the range of time when an operation can take place.

6. At this point a capacity constrained optimization is run. It will now rearrange the planned start times so as to be the soonest possible (EPST) while not exceed the capacity of the resources.

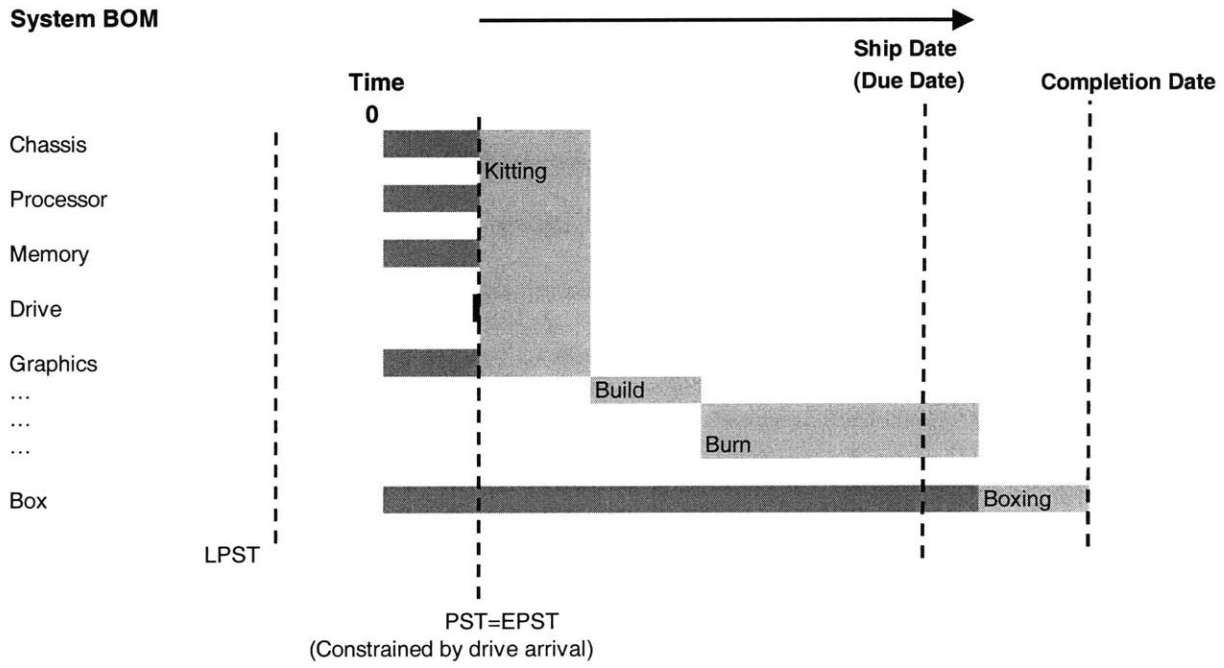


**Figure 2-4: Case 1 – PST=EPST**

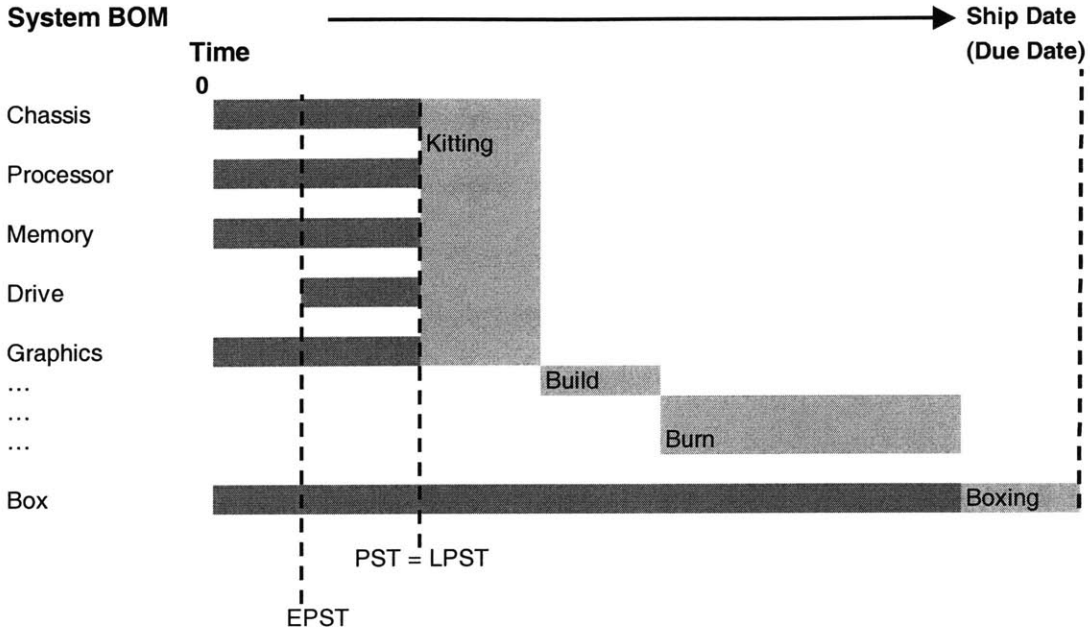


**Figure 2-5: Case 2 – PST=LPST**

7. The human scheduler reviews the material plan and the schedule. Steps 5 and 6 may be repeated. The scheduler may reassign orders to lines different than their original assignment in an effort to balance the lines and avoid unnecessary downtime. The scheduler may also modify the available capacity in light of changing circumstances on the production floor.

8. At this stage, the Planner creates the detailed order schedule files and the material requirement files and some additional post-processing is done. The schedule is fed to the different assembly lines and the material plan triggers internal and external replenishment.

## *2.3 Business Requirements*

Implementation and assimilation of the pull to order approach and the new system required process changes and enhancements in several functional areas. Among them:

- Internal processes: receiving, internal material movement and spares replenishment.
- External processes: supplier hub fulfillment and systems, inbound logistics scheduling, traditionally received parts and dock door allocation.

- Planning and scheduling: the factory Planner, IT decision support systems and pre and post processing tools.
- Information technology systems: Order Control systems interface, Inventory Control interface.
- Backlog rich environment: Since plans hold for the next couple of hours, sufficient backlog of orders must exist to maintain high utilization.
- Inventory accuracy is crucial to success – at least 95% perpetual inventory accuracy needed.
- Supplier hub delivery performance – 95% on time delivery within established planning horizon.
- Data accuracy in Inventory Control system, Order Control system and supplier hub's systems.

## 2.4 Business Case

The business case for the new system and its subsequent success story support the decision to implement it. Improvements were observed and measured in a variety of different processes and metrics.

In the traditional process the right material was not always present when an attempt was made to initiate the assembly of an order. Approximately 60% of the time, not all the right parts were in the kitting area upon "traveler pull". In this case, parts cannot be brought to the kitting area immediately and typically there is a 4 to 6 hour delay before an order can be started. Thus, it was estimated that the average order delay, due to absence of proper parts, is between 2.4 and 3.6 hours. Using the new system and processes, this figure is greatly reduced. It is estimated that such failures occur only about 10% of the time, and even then the failure is due to missing parts that are already in the facility but had not been loaded to the kitting racks. Moreover, unique premium orders, which typically failed due to missing parts in traveler pull, do not fail anymore. Overall,

the average order is estimated to be delayed only 2 minutes due to this issue. On the other hand, in a backlog starved environment there are some new causes to scheduling delays. There may be an average delay of about 45 minutes because of the frequency of the planning cycle. Material replenishment on a pull-to-order basis averages 2 hours. Delays while other scheduled orders are being built average around one hour. In total, the average total delay decreases from 4.25 hours to 3.78 hours.

Figures 2-6 and 2-7 demonstrate the improvement in the cycle time and inventory metrics following the implementation of the new software system.

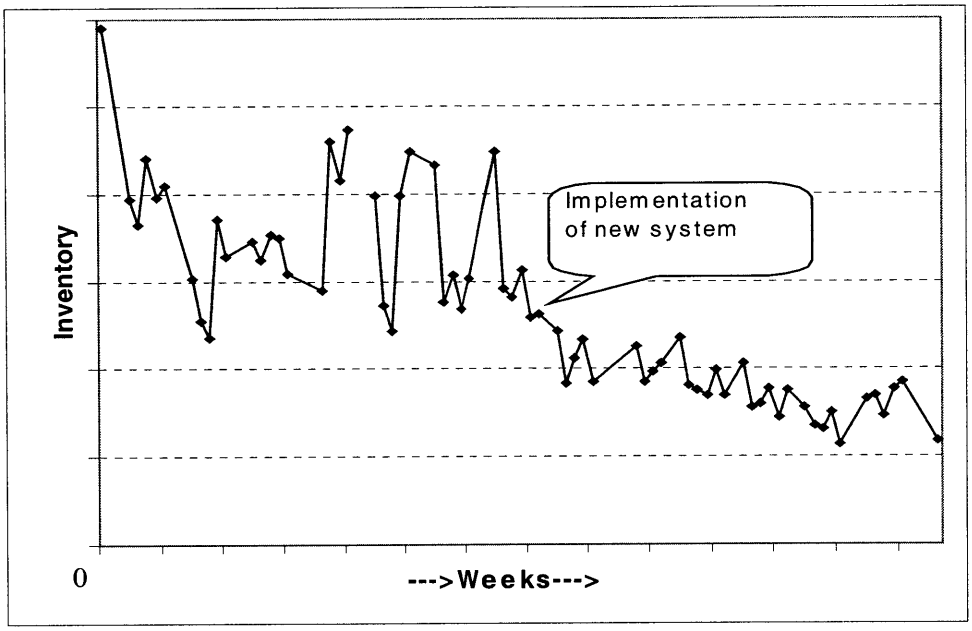


**Figure 2-6: Inventory Improvement**

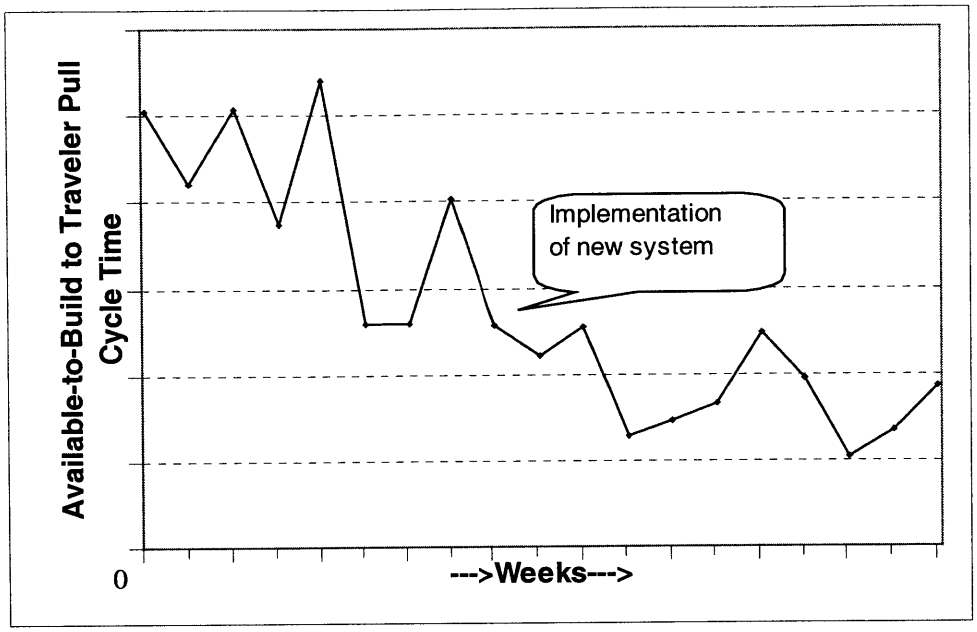


**Figure 2-7: Cycle Time Improvement**

"Dell's new approach takes the concept of just-in-time operations to new levels. The company's most efficient factories, such as an Austin plant that makes its Optiplex line of corporate PCs, order only the supplies required to keep production running for the next two hours. As the two-hour clock winds down, suppliers--who keep gear in a warehouse near Dell's factories--are electronically told what to deliver so Dell can build the next two hours' worth of computers. That virtually eliminates parts inventory ... Now, PCs often are loaded onto trucks for shipment just 15 hours after the customer clicks on the buy button, down from at least 30 hours in the past." *(David Rocks, BusinessWeek)*

# 3 Chapter Three: Problem

## *3.1 Context*

The issue of assignment of orders to assembly lines has become of greater importance during the implementation of the new system. One expects that the Planner would assign the orders to lines. After all, the existing Planner is the software module that is responsible for the actual scheduling and has access to all the information regarding capacities, priorities and material availabilities in the factory. Unfortunately, the Planner software package lacks the capability to assign orders to lines intelligently and external pre-processing is called for. That is, the assignment of orders to lines is determined exogenously and is an input to the Planner.

In the past, order assignment was executed on the basis of configuration. Each configuration – chassis, motherboard, CPU, etc. – was directed to a different line. However, the "traveler pull" was done in a "surf-to-download" pull manner. An operator would view the orders waiting to be assembled on her line and choose which of those orders to initiate. In case of uneven workload distribution among the lines the operators could manually direct certain orders to alternative assembly lines. Those lines would have to absorb the new configurations and manage any material related adjustments that might have had to be made.

The new system called for new modeling of the facilities. Each PC order had to be assigned a "routing" – a sequence of operations and resources that a PC system needs for completion. Each configuration has a predefined sequence of resources it requires, and the time it is expected to consume in each such resource. The actual routing specifies for each generic resource which physical resource, typically out of a number of alternatives, is the one scheduled to be used. However, typically Dell factories are physically designed in a way that does not allow for great flexibility in term of routing permutations. Thus, a decision on a kitting line typically dictates a choice between a small subset of building cells and burn racks and a single boxing line.

The Planner, in our case, gets the kitting and boxing resources as a given and then optimizes for other variables. In other words, the Planner pretty much treats an "assembly line" – a kitting area, several build cells, burn racks and a boxing area – independently from other assembly lines. Ideally, we would have liked the Planner to take all factors in consideration when deciding on an optimal, capacity wise and priority-wise, routing and schedule.

Line assignment, done prior to the Planner optimization, lacks complete information. By design, the external assignment process can extract as much information as deemed necessary. However, extracting all the information the Planner uses essentially replicates its functionality and may well replace it altogether. Clearly, that is not our goal in this case. A simpler, relatively straightforward, solution is called for. Figure 3-1 shows the logical flow of the assignment. Our problem is highlighted.

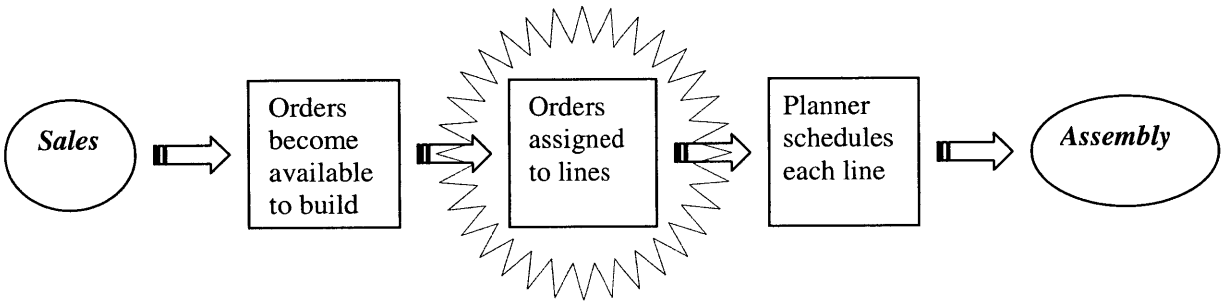


**Figure 3-1: Logical Flow**

## *3.2 Objectives*

The assignment of orders to lines is meant to meet several goals. First and foremost, there is a desire to keep the distribution of work, in terms of hours of workload, as balanced as possible. This in turn, achieves high utilization of the different resources and eliminates a fraction of the down time that can be regarded as unnecessary. Additional objectives are to maintain a low mix of configurations on each line. Building the same systems on a line is considered to be a more efficient process.

A small number of configurations equates to a small number of chassis and motherboards. Therefore, the general objective is to minimize both the number of different chassis and the number of different motherboards on a line. In some special cases, for different factories, we will want to minimize chassis and motherboards but perhaps until reaching a certain goal of, say, two chassis per line.

In summary, there are three different objectives for the assignment algorithm:

- Distribute the workload evenly among the lines.
- Minimize the number of chassis types staged on each line.
- Minimize the number of motherboards types staged on each line.

These are three uncorrelated, often independent, objectives. Typically, no one line will be the optimal assignment choice in respect to all three objectives. In order to solve this problem, a way to determine what is "optimal" must be defined and pursued. Clearly, such a definition depends upon the business logic that is behind the challenge.

- Down Time

Reduction of idle time of resources due to sub-optimally assigned work is the main driver behind the objective of a balanced workload. In many cases orders were assigned to lines based on their configuration, not accounting for workload balance. At the same time, other lines, set up to build other configurations, stood idle as no such orders were coming in. Due to the build to order nature of Dell's operations, accurate forecasting at a high time granularity is tricky to achieve. Obviously, such downtime translates into throughput loss and actual facility capacity reduction. Throughout most of a typical quarter, loss of throughput directly causes unnecessary plant and equipment costs but most significantly increases labor costs in the form of overtime pay. It is estimated that unnecessary downtime costs Dell over a million dollars a year, in the company's current manufacturing configuration.

- Variability

As a rule, operation managers at Dell prefer to build similar system configurations on the same line. It is believed, based on experience, that the more identical the systems are the less time the kit and build processes consume. No scientific survey of this phenomenon has yet been conducted; however, experience shows that after a short learning curve, the rate at which the workers build increases by around 10%. This is presumed to be due to the fact that people work faster when they do the same series of operations repeatedly. Chassis and motherboards (more precisely chassis, motherboard and CPU) determine the system configuration. Thus, minimizing the number of chassis and motherboards implicitly reduces the number of configurations being assembled. This observation defines the entire facility as a non-workload conserving system since assignment of different orders to different lines does affect the total workload. This information is not being modeled by Dell's demand fulfillment software. According to the model used, the system configuration alone dictates the expected time at each operation, hence the advantage in scheduling similar orders consecutively is not exploited.

- Other

Consideration should be given to several additional factors. In some facilities, typically those that assemble products that cater to the consumer and small business markets, the quantity of systems per order is smaller. Therefore, manual reassignment of orders, for workload balancing purposes, is quite difficult and time consuming. While reassigning a single order of 50 systems in one transaction may be an acceptable way to balance lines, it is clear that performing 50 different transaction to achieve the same outcome is cumbersome. As a result, these facilities are in greater need of a solution to the assignment problem.

The assignment problem also has some implications on the length and frequency of the planning cycle. As average manual intervention consumes more time, the average length of a planning cycle increases. Therefore, the elimination, or significant reduction, of manual

intervention in any stage in the planning process, and in the balancing stage in particular, permits the flexibility of having shorter, more frequent, planning cycles.

## 3.3 Constraints

There are several constraints that limit any solution to the assignment problem. First and foremost are the physical material constraints. One of the motives behind minimizing chassis and motherboards is the fact that there is limited physical staging area for these components on the kitting lines. Typically, up to three or four different types of chassis can be staged simultaneously at a single kitting line. Motherboards are restricted as well, although not as severely as chassis.

Another very significant constraint is time. As described in the previous chapter, Dell's planning cycle frequency, the frequency of when orders are scheduled and materials ordered from the suppliers, is very high. Due to this high frequency, and the fact that often scheduling has to be planned not precisely at the expected times, assignment of orders to lines has to be done in a near real time fashion. Therefore, the assignment process has to be initiated rather frequently in order not to be dependent on large backlogs of orders. Furthermore, the actual assignment algorithm has to be computationally efficient and to execute within a short time window during the planning cycle. An algorithm that takes more than several minutes on the available hardware is unacceptable.

Some orders require special, "premium", handling. There are two kinds of premium processes at Dell. Often, not all lines can perform one or both kinds of these special processes. Thus, typically when an order requiring these processes is waiting to be assembled, it has to be assigned to one of a smaller subset of possible lines.

Finally, there are some "organizational" issues that have to be addressed. An overly complex, incomprehensible algorithm will not serve its purpose. Eventually, scheduling managers will have to run the planning cycles and understand the implications of each step. The

manufacturing processes are not precise operations and are subject to noise. Thus, a simple but good solution is preferred over a complicated "optimal" solution.

## 3.4 Scope

The desired solution is bounded by the current high-level system architecture. Thus, the solution still has to be a "pre-Planner" algorithm and not an enhancement of the Planner itself. In addition, the solution should not essentially be a replication of the Planner. Hence, it will obviously require additional information about the state of the factory and the queues at the lines. However, it would not be desired to accumulate all the information that is essentially collected by the Planner and emulate itself. This also brings us back to the question of simplicity. Furthermore, optimization on the single system level cannot be achieved by a solution to this problem alone, since the Planner decides the actual scheduling and sequencing.

# 4 Chapter Four: Solution

## *4.1 Existing Situation*

The existing solution for the line assignment problem is a manual configuration based proportional system to line allocation. As mentioned earlier, the initial assignment logic was exclusively based on system configuration, i.e. systems of a specific configuration would be directed to one specific line. This was obviously a very crude method and necessitated frequent manual intervention by the operators. This solution has evolved into a configuration based proportional assignment (see Figure 4-1). Each configuration is manually assigned to one or more lines, in pre-defined proportions. These proportions are determined in a manner that balances the "forecasted" load for each line. The forecasted load is based on quarterly demand predictions. The assignment of configuration to lines is decided in a manner that meets chassis and motherboard constraints.

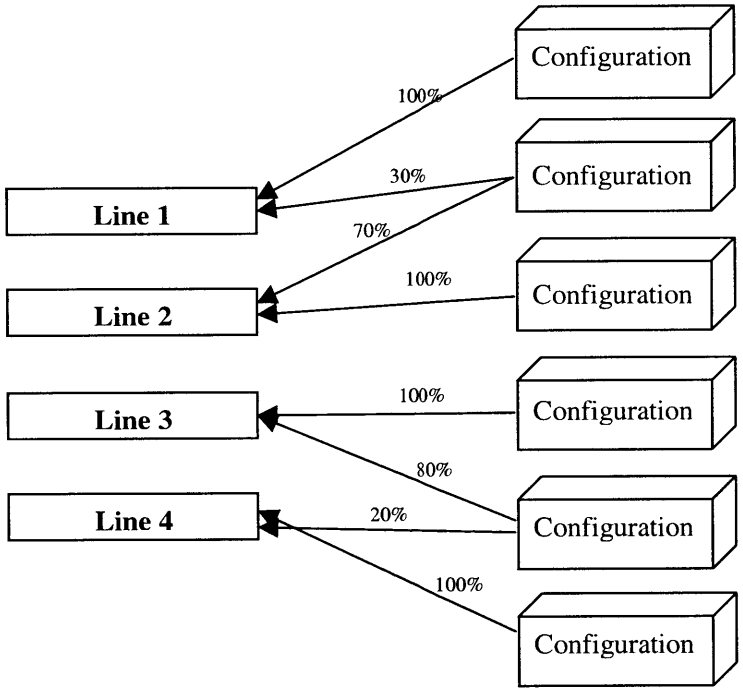


**Figure 4-1: Configuration Based Proportional Assignment**

Dell has developed a spreadsheet tool to help the schedulers determine the matching of configurations to lines. The scheduler sees the forecasts, in daily demand terms, and decides on the proportion of each configuration that will be directed to each line. Consequently, the scheduler sees the chassis and motherboard requirements that are derived from her decisions.

There are several noteworthy drawbacks to this approach. First, this approach is based entirely on the forecasted numbers and not on the actual situation on the factory floor. Furthermore, it disregards the characteristics of the actual backlog. Thus, adjustments to the percentages need to be updated frequently. Second, this process is manual. Still, a scheduler has to manually decide, perhaps with the aid of tools, which configurations are assigned to which lines. This is costly in terms of the time a scheduler has to devote to this scheme. Clearly, the scheduler has the flexibility to update this assignment as frequently or infrequently as she pleases. However, the more infrequently she does so, the more difficult it is to maintain a balanced workload. This leads to the third issue – the percentages are based on quarterly forecasts. The accuracy of these forecasts is arguable at best, but when daily forecasts are derived from these figures their accuracy is even more doubtful due to the variable demand nature of Dell's business. In conclusion, this scheme requires a high degree of manual intervention and adjustments are usually made reactively, at a considerable delay.

In the following sections we shall describe various attempts to provide an answer to the line assignment problem. Since this paper discusses a real life project, the evolution of the solutions suggested coincides with the evolution of the definition of the problem. Stricter and more relaxed constraints appeared as the decision makers grew to understand the intricacies of the problem and articulated exactly what is required by the various Dell facilities.

## *4.2 Integer Programming Solutions*

Our first approach towards creating a better solution for this problem involves Integer Programming optimization. As stated before, orders are typically assigned in batches of ten

minutes or so. As the data suggests, typically around 100 systems are to be assigned in one batch. By formulating an offline optimization problem we can observe the state of the queues of each line, as well as the types of configurations waiting to be scheduled before we reach an "optimal decision". By defining clear objective functions and constraints we are able to determine the "percentages" proactively in greater accuracy, based on the actual situation.

### 4.2.1 Integer Program with Priorities

The first attempt to suggest an improved solution is based on imitation of what the schedulers are actually doing. The schedulers are in essence deciding on a priority order in which they want the systems assigned. They prefer that all systems go to a specific line, yet due to workload distribution consideration they accept that some orders go to alternative lines, as a lower priority. Therefore we devised the following program:

Parameters:

| | |
|---|---|
| $N$ | - Number of different configurations |
| $n_i \quad i = 1..N$ | - Number of orders of configuration i |
| $q_{il} \quad i = 1..N, l = 1..n_i$ | - Quantity per order |
| $M$ | - Number of lines in the factory |
| $l_j \quad j = 1..M$ | - Workload of each line (before the assignment) |
| $r_j \quad j = 1..M$ | - Production rate of each line |
| $p_{ij} \quad i = 1..N, j = 1..M$ | - Priority for each configuration and line combination |
| $w$ | - "Importance" of priorities in respect to balanced lines. |

Decision Variables:

$X_{ilj} \quad i = 1..N, l = 1..n_i, \; j = 1..M$ - $X_{ilj} = 1$ if order i,l is assigned to line j, 0 otherwise.

Objective:

We shall use an auxiliary variable to simplify the objective function.

$$F_j = \frac{l_j + \sum_{i=1}^{N} \sum_{l=1}^{n_i} q_{il} X_{ilj}}{r_j}$$ - Load factor for line j

$$\text{Minimize} \sum_{j=1}^{M} \left( \mathrm{F_j} - \frac{1}{M} \sum_{j=1}^{M} F_j \right)^2 + w \sum_{i=1}^{N} \sum_{j=1}^{M} p_{ij} \sum_{l=1}^{n_i} X_{ilj}$$

Constraints:

$$\sum_{j=1}^{M} X_{ilj} = 1, \quad i = 1..N, l = 1..n_i$$ - All orders have been assigned to a line

$X_{ilj} \in \{0,1\} \quad i = 1..N, l = 1..n_i, \; j = 1..M$ - Binary decision variables

Several elements in this program are worth mentioning. The objective function has two distinct components: the "balance" component and the "priority" component. These objectives are independent from each other. Thus, an arbitrary weight *w* determines the relationship between these two components. We chose to ignore "premium" service constraints at this stage.

Each configuration-line pair is given a priority in a manner that reflects the scheduler's preferences. For example, if the scheduler decides that configuration A is to be assigned to line 2 as a first priority and to lines 3 and 4 as a second priority, she will set $P_{A2} = 1$, $P_{A1} = 2$, $P_{A3} = 2$. If there are lines where she does not want to assign configuration A under any circumstances, she will set $P_{Ak} = A$, k being the other line and A being a very large number. It can be seen that the scale of these priorities is arbitrary as well as the cardinal differences between consecutive ordinal rankings.

The first component of the objective function is a quadratic formula. It tries to minimize the sum of the squares of the differences between the load factors and a mean load factor. Our goal is to create a condition where the variance of the load factors is minimal. In the same spirit, it seems to be reasonable to include a quadratic "penalty" to minimize the amount of imbalance.

The introduction of a quadratic objective function requires more computationally complex algorithms. While linear programs are readily solved by widely available software packages, nonlinear programs, although being solved as well by commercial software, necessitate more time computationally. An alternative approach would be to use a linear objective function that attempts merely to minimize the maximum load factor, or minimize the difference between the line with the highest load factor and the line that has the least load factor:

1) Minimize $Y + w\sum_{i=1}^{N}\sum_{j=1}^{M} P_{ij}\sum_{l=1}^{n_i} X_{ilj}$

s.t.

$Y \geq F_j, j = 1..M$ - Y is the maximal load factor

or

2) Minimize $(Y - Z) + w\sum_{i=1}^{N}\sum_{j=1}^{M} P_{ij}\sum_{l=1}^{n_i} X_{ilj}$

s.t.

$Y \geq F_j, j = 1..M$ - Y is the maximal load factor

$Z \leq F_j, j = 1..M$ - Z is the minimal load factor

The second option is viewed as more desirable. For example, assume that an order is not assigned to the line with the highest workload, as we hope. We still prefer that it will be assigned to the line with the least work and not just to any line arbitrarily.

### 4.2.2 Integer Program with No Priorities

At this stage we eliminate the need for priorities. The user of the previous program still needs to intelligently prioritize the lines. A simpler approach is to have the user input the actual constraints, and have the program "deduce" the priorities for her. Therefore, our next approach is to define the actual constraints. In this case we referred specifically to the objective of not having more than three types of chassis staged on one line at the same time. We also included the two "premium" process constraints.

Parameters:

$N$ - Number of orders

$q_i \quad i = 1..N$ - Quantity per order

$p_i^d \quad i = 1..N, d = 1..2$ - "Premium" process flags for each order, $p_i^d = 1$, if order $i$ is of premium service $d$, 0 otherwise.

$P^d \quad d = 1..2$ - Set of lines capable of "premium" process of type d.

$M$ - Number of lines in the factory

$T$ - Number of different types of chassis

$c_{ik} \quad i = 1..N, k = 1..T$ - $c_{ik} = 1$ if order i is of type k, 0 otherwise.

$s_{jk} \quad j = 1..M, k = 1..T$ - $s_{jk} = 1$ if type k is already scheduled for line j, 0 otherwise.

$l_k \quad k = 1..M$ - Workload of each line

$r_k \quad k = 1..M$ - Production rate of each line

Decision Variables:

$X_{ij} \quad i = 1..N, j = 1..M$ - $X_{ij} = 1$ if order i is assigned to line j, 0 otherwise.

$Y_{jk} \quad j = 1..M, k = 1..T$ - $Y_{jk} = 1$ if type k is being scheduled to line j, 0 otherwise.

Objective:

$$F_j = \frac{l_j + \sum_{i=1}^{N} q_i X_{ij}}{r_j}$$ - Load factor for line j

$$\text{Minimize} \sum_{j=1}^{M} \left( \mathrm{F_j} - \frac{1}{M} \sum_{j=1}^{M} F_j \right)^2$$

Constraints:

$$\sum_{j=1}^{M} X_{ij} = 1, \quad i = 1..N$$ - All orders have been assigned

$$1 - p_i^d + p_i^d \sum_{j \in P^d} X_{ij} = 1, \quad i = 1..N, \quad d = 1..2$$ - "Premium" process constraints

$$\sum_{i=1}^{n} c_{ik} X_{ij} + s_{jk} \le K \cdot Y_{jk}$$ - Chassis usage indicator

$$\sum_{k=1}^{P} Y_{jk} \le 3, \quad j = 1..M$$ - Chassis constraint

$$X_{ij}, Y_{jk} \in \{0,1\} \quad i = 1..N, j = 1..M, k = 1..P$$ - Binary decision variables

In this program, we use the same "balance element" for the objective function. Two new constraints are introduced. The first is an "if then" constraint – if the premium flag is on then the order should be assigned to one of the acceptable lines. The second is the chassis constraint – with its indicator constraint. K, a very large number, together with this constraint causes Y to be an indicator of whether a chassis is assigned top be staged on a line (unless not a binding constraint). In the same manner, motherboard constraints can be added.

The main obstacle encountered with this approach is computation time. General algorithms to solve Integer Programming problems may take an unreliably long time – a clear violation of one of the major practical constraints. In addition, as the problem is articulated more precisely, the number of variables increases and the solution space expands dramatically.

## *4.3 Online Heuristics*

### 4.3.1 Known Approaches

Due to the computational obstacles encountered with the Integer Programming optimization approach, we investigated a more computationally efficient heuristic approach. The motivation is that our real goal is to have a good, acceptable solution rather than an absolute "optimal" solution, whatever "optimal" may be. Since the entire assembly process is subject to considerable external noise in every step of its way, the disparity between a good solution and an "optimal" solution is not significant in our context.

One of the main differences between the heuristic algorithms and Integer Programming is the elements of the system being examined. In the Integer Programming approach we looked at the state of the system (queues, chassis, etc.) as well as the entire batch of unassigned orders. Thus, we looked upstream into our entire set of waiting orders. However, the heuristic approach explored is an example of an online algorithm, an algorithm that does not know what the entire input will look like.

The intricacy of our problem is due to the three independent objectives we wish to optimize simultaneously. The main objective, and the trigger to this entire effort, is the desire to balance the workload and to prevent unnecessary downtime. Again, this must be achieved in parallel to meeting requirements regarding chassis and motherboards, to minimize their number on each kitting line. These objectives typically do not coincide. Choosing one line over another will ordinarily improve our position in respect to one objective yet compromise the other.

Many real life problems involve multiple objectives, which must be optimized simultaneously. In many cases, single objectives may be optimized separately from each other and an understanding can be acquired as to what can be achieved in each specific dimension. However, an optimal result for one objective typically implies low performance in one or several of the other objectives, creating the need for some kind of a compromise. An "acceptable" solution to these problems is often sub-optimal in the single objective sense and is subjective by nature.

Simultaneous optimization of multiple objectives rarely generates a single, perfect optimal solution as in the case of optimization of a single objective. Multiobjective optimization tends to come up with a set of alternatives, all considered "equivalent" unless specific information is given as to the "importance" of each objective relative to the others. As the number of competing objectives increases, the problem of finding an acceptable compromise quickly becomes more and more complex.

There are no magic solutions to multiobjective optimization problems. Other than plainly listing an order of all possible (discrete) rankings, the literature[1] mentions three basic approaches to address these problems. The first is to use a lexicographical (sequential) optimization scheme. Thus, we decide on an order of the multiple objectives and optimize them one by one based on this order. After the first iteration we are left with only the alternatives that are optimal for the first, apparently most important, objective and among those we proceed to a second iteration based on the second objective, and so forth.

The second approach is aggregation to a single objective function. This approach requires that each dimension have a cardinal metric in which alternatives are ranked. To assess the optimum, each set of multidimensional metrics is combined to a single metric, a function of the individual independent metrics. Weighted average and aggregation of penalty functions are

---

[1] Keeney & Raiffa: *Decisions with Multiple Objectives,* New York: John Wiley & Sons, 1976 and Hwang & Masud: *Multiple Objective Decision Making,* Berlin: Springer-Verlag, 1979.

commonly used. Other formulas, such as those that translate the original metrics into terms of utility, are also widespread.

The last commonly recognized approach is goal programming. Using this approach, we set goals for each objective and try to attain all of them simultaneously. Once a goal is met on one dimension, we are (at least initially) indifferent between all the alternatives that have met that goal. Hopefully, we will be left with one or several alternatives that have attained all goals; otherwise we will have to relax some of our goals. A variation is the goal attainment method, where the aim is to minimize weighted difference between objective values and corresponding goals.

More sophisticated approaches for complex multiobjective problems have been developed. Genetic and evolutionary algorithms[2] attempt to imitate natural phenomena of concurrent optimization on several dimensions. In addition, algorithms with randomized elements have tried to attack this problem from a different angle.

We have decided to come up with an algorithm that is essentially a combination of several aforementioned approaches in accordance with the business logic expressed by the decision makers. This problem does not call for complex cutting edge algorithms but rather for a relatively simple solution that can be easily comprehendible, and provide a good response for most prevailing cases. As a (lexicographically) first screen, we suggest evaluating the workload factor. The alternatives that pass this screen are examined using a discrete preference order as well as a weighted average approach.

---

[2] Fonesca, Carlos M., Fleming, Peter J.: *An Overview of Evolutionary Algorithms in Multiobjective Optimization,* Evolutionary Computation, 3(1): 1-16, spring 1995.

### 4.3.2 Proposed Solution

In order to maintain generality, we refer to chassis and motherboards as components. Component types refer to the different types of chassis or different types of motherboards. In addition to what was mentioned earlier, more accurate workload calculations become possible as the data for expected time per operation per system becomes available. Thus, workload is calculated as the actual time consumed at the kitting operation by all the systems, taking into account that some configurations may consume more time than others. Furthermore, considerations such as planned downtime (e.g. for maintenance) are also combined into the workload calculation.

The general structure of the proposed solution involved two phases: a construction phase and an improvement phase. The input to the solution is a batch of orders that have accumulated. During the construction phase we will iterate through the batch of orders one at a time and assign each order to a line. The construction phase consists of three main steps (as shown in Figure 4-2). During each step the set of eligible candidate lines, initially the entire set of lines, may only decrease in size. Thus, eventually we will end up with one alternative. The first step eliminates obvious bad choices for assignment – lines that are not an option due to hard and binding constraints. The second step selects only lines that are within a certain time window – lines that are the least loaded. The third step selects the line that has the best fit in terms of the component types. During the following phase, the improvement phase, the algorithm tries to improve the assignments by trying to move orders to better fitting lines.

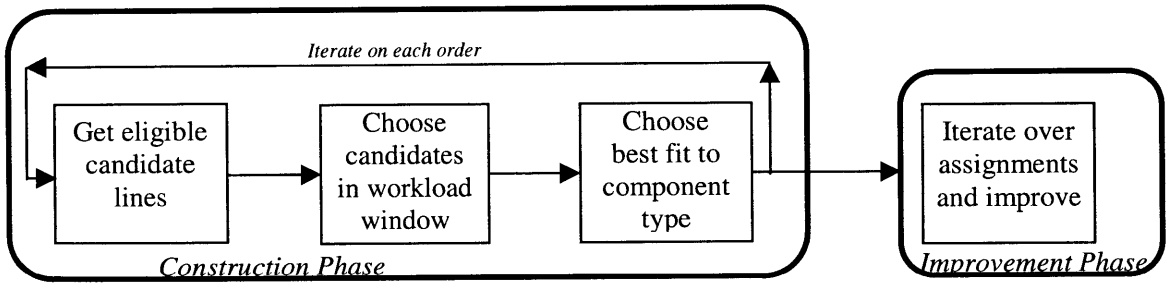


**Figure 4-2: Overview of Solution**

- ***Construction Phase***

In the construction phase, we assign orders one by one starting with high quantity orders first. It may be easier to assign large quantities earlier on, when the lines are most unbalanced.

As a first step when attempting to assign an order to a line we disqualify all those lines that are infeasible upfront. These lines are bound by such hard constraints as the ones dictated by the orders that require "premium" operations and constraints on upper limits of physical capacities of staging areas of lines. Typically, a line will not be able to stage more than a few types of each component in its kitting area.

In the second step, while looking solely at the workload factor, we are using a variation of the goal attainment approach to designate a window of acceptable alternatives. The considered window always begins with least loaded line, which has workload $x$, and extends to $y$, which is determined as follows:

$$y = \max(s, cx)$$

where $s$ is the "starvation" threshold, in hours, and $c$ is the coefficient of tolerance. A typical graph is shown in the figure:

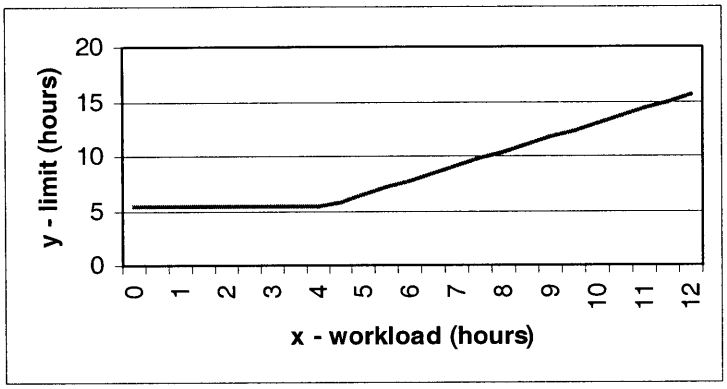


**Figure 4-3: Graph of the Window Limit (s = 5, c = 1.3)**

For example, assume the following workloads in a facility with eight lines:

| Line | Workload (in hours) |
|---|---|
| 1 | 6 |
| 2 | 10 |
| 3 | 7.5 |
| 4 | 6.5 |
| 5 | 7 |
| 6 | 9 |
| 7 | 9.5 |
| 8 | 8 |

Suppose the starvation threshold is 5 hours and the coefficient $c$ is 1.3. The window in the example extends from 6 hours, the minimal workload, to 7.8 hours. Therefore, only lines 1, 3, 4 and 5 are within this workload window and remain eligible.

After reducing the number of alternatives, the last criterion is the already staged components. Here we use a combination of a simple ordinal ranking and a weighted average. We define "classes" of alternatives for each component according to two parameters: the existence of the specific component type that corresponds to the evaluated configuration and the number of components types already staged. We also introduce a metric - the percentage of the evaluated components of all the staged components (normalized for time consumption of each system). Assume A is the evaluated component:

Class I: all lines that have an A staged.

Class II: all lines with one component type staged – not A.

Class III: all lines with two component types staged – both not A.

and so forth.

| Workload already assigned to line | Percentage of each chassis | Metric *c* for assignment of chassis A | Class |
|---|---|---|---|
| 400 | A – 200, 50%<br>B – 100, 25%<br>C – 100, 25% | 50% | I |
| 350 | A – 350, 100% | 100% | I |
| 300 | B – 270, 90%<br>C – 30, 10% | 0% | II |
| 2000 | A – 400, 20%<br>B – 1600, 80% | 20% | I |
| 150 | B – 150, 100% | 0% | II |

**Table 4-1: Chassis Classification Example**

Table 4-1 shows an example of classification for a single component. For example, in the first row, 200 out of 400 minutes of the backlog are due to type A systems. Therefore, since A is already staged, this is class I and the percentage is 50.

In order to aggregate the information about classes for all components we use an aggregation function. The function chosen by Dell is sum of the two classes (choose the minimum). In case of a tie, choose the alternative with the lowest chassis class. An additional method to express the same idea is to use a weighted average formula such as:

$$R = w_1 C_1 + w_2 C_2$$

where $C_1$ and $C_2$ are the respective component classes and $w_i$ are predetermined weight. The order of preferences is the sorted list according to R.

| Chassis / MB | I | II | III | ... |
|---|---|---|---|---|
| **I** | 1 | 2 | 4 | 7 |
| **II** | 3 | 5 | 8 | |
| **III** | 6 | 9 | | |
| **...** | 10 | | | |

**Table 4-2: Order Chassis / Motherboard Preferences**

Since we hope that we will be assigning orders to Class I lines, both for chassis and motherboards, we determined a mechanism to choose among several alternatives that fit this

category. In this case we again use the weighted average approach using the percentage metric to aggregate the information to one data point regarding both dimensions. We use $S$:

$$S = u_1 c_1 + u_2 c_2$$

where $c_1$ *and* $c_2$ are the percentages of the respective component types on the line (again, normalized for time consumption of each system). The weights $u_i$ defines the relative "importance" of the types. Assuming chassis is slightly more important $u_1$ will be 1 while $u_2$ will typically be around 0.7 for motherboards. The tiebreaker between alternatives in the same class coordinates but not in class I of both is simply the lower workload.

There are several motivations for choosing this heuristic. During the first step we eliminate lines that fail to meet physical deterministic constraints. Such orders cannot be assigned to lines that physically are incapable of performing such operations. One of the constraints mentioned was the limit of the physical space available for staging different component types (such as chassis). However, under certain conditions an order could theoretically exceed such a limit if no other alternative were available (e.g. the order requires premium service although it has a different chassis type).

The most important objective, motivating our entire research is the need to pursue a balanced workload and to eliminate "starving" lines. After every planning cycle there are materials for a certain time horizon, typically several hours. A line that has work for less than the "starvation threshold", the length of the planning cycle, will run out of orders and starve. The business decision makers are most concerned about this scenario. Such a situation means that a line will incur downtime. If there were enough orders waiting elsewhere, this down time is not necessary.

Any set of production lines must have some pre-planned amount of idle time in order not to have infinite queues. From a queueing theory perspective, one cannot get away from starvation time in a stable queue environment. Even in an unbalanced system, if the queues remain finite,

they will eventually recycle to zero length and the total throughput will be that which is desired. There will be a difference between the *average* throughput and the *maximum* capacity on all lines regardless of the control policy.

An "ideological" issue arises when dealing with assignment to soon-to-be-starving lines - whether to balance among the starving lines themselves. The same amount of downtime, and therefore the same financial loss, will be incurred whether one line is down for three hours or two lines are down, one for two hours and the other for one hour. The implications of such a decision are more organizational in nature. How would busy workers on one line feel when their friends on the neighboring line are idle? Does it contradict the company policy of measuring each line manager on his or her line attainment? The decision makers decided to deal with the softer issues rather than compromise on the chassis and motherboard front. Thus, "starving" lines are seen as equivalent for assignment purposes despite the possible variability in their workloads.

When dealing with the component type "fit", we are willing to consider all reasonable alternatives, i.e. those that have the lowest workload or close to it. We do not want to introduce a new component type to a line when another line, perhaps slightly more heavily loaded, corresponds perfectly in terms of staged components. In addition, as the workload gets higher, in absolute terms, and farther from the starvation point, we are more tolerant toward considering alternatives that are sub-optimal workload-wise.

Clearly we prefer to assign orders to Class I lines in order to avoid introduction of new component types to lines (this is true for both chassis and motherboards). In the case of two constraints we can look at the two dimensional matrix of classes. The question that is now asked is how to ordinally rank the class coordinates. The business decision makers detailed their preference order although there was some uncertainty about some alternatives. However, a general pattern emerged: the classes should be ranked according to the sum of the classes, with chassis being a tiebreaker. This pattern is portrayed in the Table 4-2. The extreme points, i.e. the

coordinates where (at least) one of the classes meets a physical constraint, may be different than what is portrayed in the table.

In the case that we have to decide among class I alternatives, we will prefer to assign chassis A to a line that has the highest percentage of A's already. Thus, we will be "driving out" types that are in minority.

- ***Improvement Phase***

As discussed, the online algorithm approach fails to leverage the information embodied by the entirety of orders in a batch. Unlike the Integer Programming approach, an online algorithm assigns orders one by one. We therefore suggest an enhancement to the algorithm that takes some advantage of the available information. We propose adding to the aforementioned "construction" phase, which assigns the orders as described, an "improvement" phase that reassigns some of the orders.

The improvement phase attempts to reassign orders to more suitable lines. The objective is to improve the initial assignments by making appropriate switches. First, improve component type assignments. Second, improve load balancing (opposite to the sequence in the construction phase). We go over assigned orders, always examining an order assigned to the line with the maximal workload. In either case, whether that order was switched or not we proceed to the next order on the line with the maximal workload or, in case we examined the last order of that line, to the next heaviest loaded line. For each order execute the following instructions (assume the current order is assigned to line *y*):

1. Determine a ranking order on the lines (in terms of component types) after the construction phase. These ranking are the same as the ones used to choose among different classes.

2. Choose the line *x* with the best overall ranking (as long as its ranking is higher than that of line *y*):

If line *x* (after the addition of the order) has a smaller load than *y* (including the order) - move the order to line *x*.

If the statement is false try the line with the next highest rank after *x*.

The algorithm is illustrated in Figure 4-4:

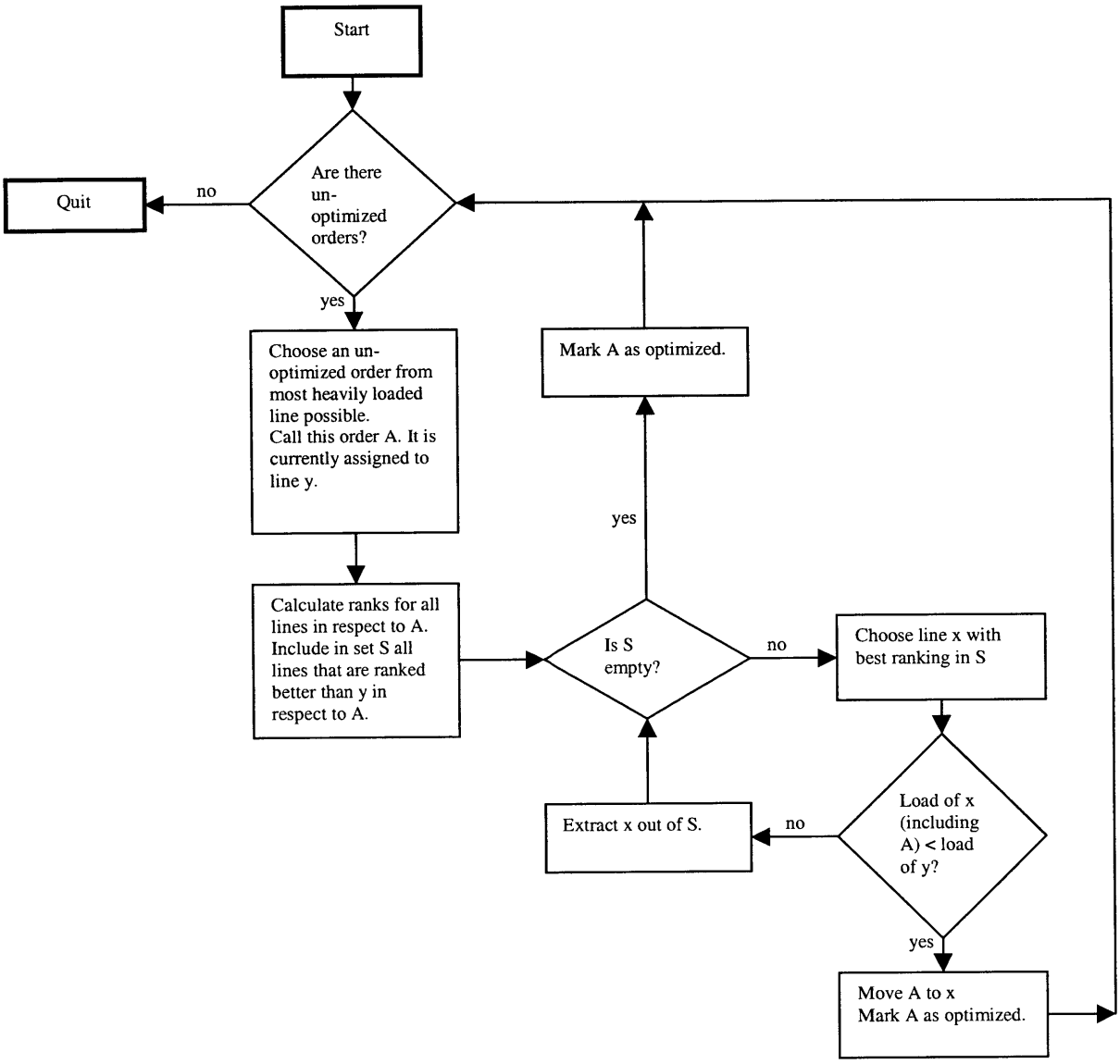


**Figure 4-4: Flow Chart of Improvement Phase**

### 4.3.3 Generalization of Objectives

A different flavor of the algorithm, dealing differently with the component objectives is required by some of the facilities. The objective in this case is not necessarily to minimize chassis and motherboards simultaneously down to one, but rather to minimize them down to a certain target and strive to maintain that target. Moreover, in some cases only optimization of one dimension (chassis or motherboards) is necessary due to the lack of any practical constraints on the other.

By minimizing a component type toward a target $T$, $T > 1$, we mean that we prefer having $T$ types staged on a line over both $T-1$ and $T+1$. The motivation behind this approach is that some production strategies prefer having $T$ types of systems being assembled on the same line in order to avoid idle time in case one of the types is put on hold (due to reasons such as unresolved engineering issues).

Our approach in this case is very similar to the previous one. We define the classes slightly differently. Assume A is being evaluated. The classes are defined as follows:

Class I: all lines that have A staged and no more than $T$ types

or

all lines that do not have A staged but have less than $T$ types.

Class II: all lines that have A staged but more than $T$ types.

Class II: all lines with $T$ types staged – all not A.

Class III: all lines with $T+1$ types staged – all not A.

and so forth.

In these cases, the smaller the percentage of A's is among the chassis the more attractive it is for assignment. Therefore when we use our weighted average formula we will use such a percentage multiplied by (-1).

### 4.3.4 Limits and Targets per Line

Two additional enhancements are added to the algorithm. The first is more flexible treatment of the limit on staging capacity of components. The second is having minimization targets (and maximum limits) set on the line level and not the facility level. The algorithm, as described in section 4.3.2, eliminates all candidate lines that exceed their staging capacity very early on. In fact, such constraints can only be violated if an order can be assigned to no other line without surpassing these limits or violating strict constraints such as "premium" service requirements. This enhancement ranks the lines on their "distance" from the limit and uses that criterion for an early selection. Assume order with component of type A is evaluated. Two sets of classes are defined as follows:

Max Class (the maximum limit is *M*):

Class I: all lines that have A staged and *M* types

or

all lines that have less than *M* components staged.

Class II: all lines that have A staged but more than *M* types.

Class III: all lines that have *M* types staged – no A.

Class IV: all lines that have *M+1* types staged – no A.

and so forth.

Target Class (the target is *T*):

Class I: all lines that have A staged and *T* types

or

all lines that have less than *T* components staged.

Class II: all lines that have A staged but more than *T* types.

Class III: all lines that have *T* types staged – no A.

Class IV: all lines that have *T+1* types staged – no A.

and so forth.

The algorithm now looks at the sum (or some other weighted average) of each of the two Max Classes or two Target Classes (in the case of optimization of two component types). It chooses all the alternatives in the group of lines that has the lowest sum of Max Classes and among those all the candidates that have the lowest sum of Target Classes. An additional business logic decision is to disregard the Max classes in the case that our initial workload window begins below the starvation threshold. This decision is made, again, to ensure feeding of starving lines.

# 5 Chapter Five: Analysis of Solution Approach

## *5.1 Analysis of Construction Phase*

The heuristic proposed in the previous chapter attempts to provide a solution to the multiobjective problem facing Dell. As described, defining an "optimal" desired outcome is not straightforward. Greater emphasis on one objective may be viewed by one decision maker as "better" than by some other decision maker, who prefers a different objective. Therefore, it is impractical to demonstrate that one solution is "optimal" or "better" in objective, absolute terms.

As a result, not much can be decisively argued as for the "optimality" of the solution. All that can be reasonably demonstrated is that only "pareto efficient" alternatives are chosen and that the algorithm achieves its stated goals. Pareto efficiency implies that no alternative that is inferior to some other alternative on all dimensions is chosen. We will demonstrate that the greedy decisions made at each stage follow the business logic and ensure Pareto optimality. We will examine the behavior of the algorithm in various cases.

- Pareto Efficiency

We argue here that only Pareto efficient alternatives are chosen. First, only candidates that are inside the eligible workload window are chosen. This window starts at the line with least workload and extends over a "reasonable" horizon. No line with a workload outside (greater than) this window may be chosen and therefore the chosen candidate will be from the "best" group in terms of workload. We later choose the candidates with the best fit in terms of meeting the maximum limits requirements and the best fit in terms of the component targets. By always choosing the "best" subset in respect to some objective in a greedy manner we guarantee that the alternative will be on the Pareto efficiency frontier, as it will not be dominated by another alternative on all dimensions.

- Starvation

The primary overall objective is to eliminate situations of "starvation" on any of the lines. Eliminating such situations translates directly in to cost savings, as mentioned in chapter 3. See Figure 5-1 for example. Assume 500 is the starvation threshold. Distribution of workload as depicted in the figure is frequently observed in the field.

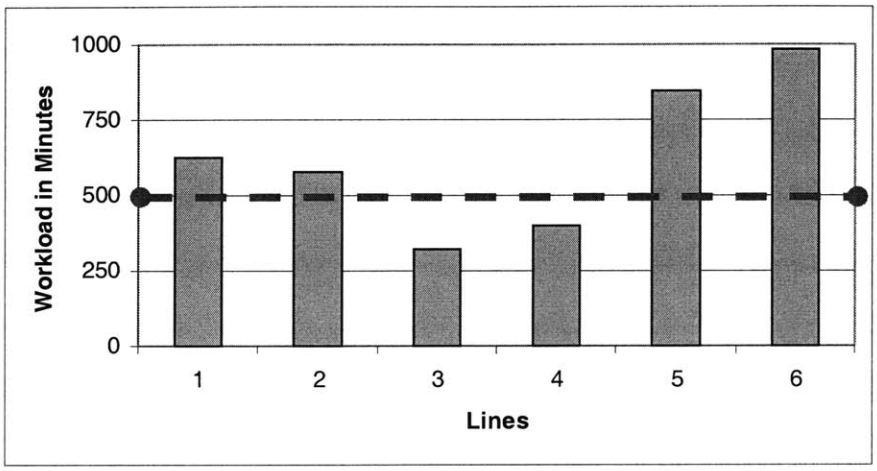


**Figure 5-1: Workload Distribution**

In the case of percentage driven assignment of systems to lines, we will find, more often than not, that some lines (lines 3 and 4 in the example) may still be under the starvation threshold due to the variable nature of the demand. This in turn will cause downtime, which could be eliminated with dynamic assignment decisions. The greedy assignment algorithm will always feed starving lines by design. Meeting component targets may be compromised, but this compromise is tolerated by the decision makers.

Under desirable backlog rich conditions, the importance of the workload balancing objective diminishes. Once the starvation threshold is a safe distance away, a larger workload window is tolerated and thus all lines are practically evaluated in terms of their maximum limits and component type targets.

Analysis of historical data reveals that around 2% of total assembly time is downtime caused by unbalanced assignments of lines. This assignment algorithm should eliminate this fraction.

- Maximums and Target

Once the workload issue is considered, the rest of the assignment is based entirely on the component type situation on the lines. Since the maximum limit is never less than the target set for the line, we have six possible parameter combinations for a line:

| Case | Maximum | Target | Example |
|---|---|---|---|
| 1 | Less than limit (Class I) | Less than target (Class I) | Max=3<br>Target=2<br><br>A – 100% |
| 2 | Less than limit (Class I) | Exactly at target (Class I) | Max=3<br>Target=2<br><br>A – 60%<br>B – 40% |
| 3 | Less than limit (Class I) | More than target (Class II) | Max=3<br>Target=1<br><br>A – 70%<br>B – 30% |
| 4 | Exactly at limit (Class I) | Exactly at target (Class I) | Max=2<br>Target=2<br><br>A – 65 %<br>B – 35% |
| 5 | Exactly at limit (Class I) | More than target (Class II) | Max=3<br>Target=2<br><br>A – 80%<br>B – 10 %<br>C – 10% |
| 6 | More than limit (Class II) | More than target (Class II) | Max=2<br>Target=1<br><br>A – 40%<br>B – 30%<br>C – 20% |

**Table 5-1: Parameter Combinations for Lines**

If we want to assign a type A system we will prefer, in terms of maximum class, all cases 1 through 5 (class I) and then case 6 (class II). In terms of target class, we will prefer cases 1, 2 and 4 (class I) over 3, 5 and 6 (class II). The maximum class consideration has lexicographic precedence over target class consideration according to the business logic. Therefore, each can be examined independently. The consideration here is relatively clear – we will rather not violate the maximum limits and if we must we will rather not introduce new types to the line (cause even greater damage). As a last resort, we will try to assign to the line closest to its limit.

In comparing this approach to the current percentage based assignment, we find that our recommendation is always poorer in this criterion. Clearly, in the current situation these "limits" are never violated since there are no such limits. Only certain component types are assigned to each line thus eliminating the need to worry about such issues. However, our suggested algorithm does cause divergence from the desired component mix in cases of workload imbalance. This is especially true under the threat of starvation.

The "class" system extends to more than one component type. The aggregation function chosen by the decision makers is the sum of the multiple classes. This formula may be easily modified if so desired. In our case chassis, the first component type, was deemed slightly more important than the motherboards, the second component type.

### 5.1.1 Simulations

We have used a simulation to assess the effects of this algorithm. The demand for this simulation is based on actual demand orders of more than a month, 5911 data points in all. The program simulates a facility with five assembly lines and a variable assembly rate at each line. We examine the distribution of component types at the assembly lines during this simulation. The code used to write this simulation used the following parameters for experimental purposes:

- Starvation threshold of 30 units.
- All units require the same assembly time.

- Facility is workload conserving, i.e. there is no gain to the assembly rate by assembling consecutive identical systems.
- Facility with 5 lines.
- No maximum limit of number of component types.
- Target of 1 on all lines.
- Based on actual data of 10 minute batches (5911 batches).
- 4 different chassis – L, M, S, T.
- 5 different motherboards – B, J, JA, S, T.
- Assembly rate per inter-batch period is 5 units + a number of units uniformly distributed between 0 and 10. This imitates the rate variability of the system.

Simulations show that the average number of chassis and motherboards increases before the starvation threshold is passed. Typically, the algorithm will tend to make more compromises while the eligible window is still below the starvation threshold. After the system as a whole crosses the starvation threshold, it tends to stabilize on a smaller average number of chassis and motherboards. Figures 5-2 and 5-3 show the result of a typical simulation based on data for a single day. Figure 5-2 shows the average number of chassis and motherboards per line. Figure 5-3 shows the average and minimum workload of all lines at the corresponding time. In this example, the average number of chassis rises up to 3.4 just as the threshold is crossed and then decreases to around 2. On the other hand, the average number of motherboards remains high. This could be due, in part, to the fact that this simulation gives greater importance to chassis over motherboards. For comparison, in the percentage based assignment method, the average motherboards per line is usually around 3, while the number of chassis was around 2.5. This empirically shows that the suggested algorithm maintains the same level of chassis and motherboards on the lines, while, at the same time, looking after the balance issue.

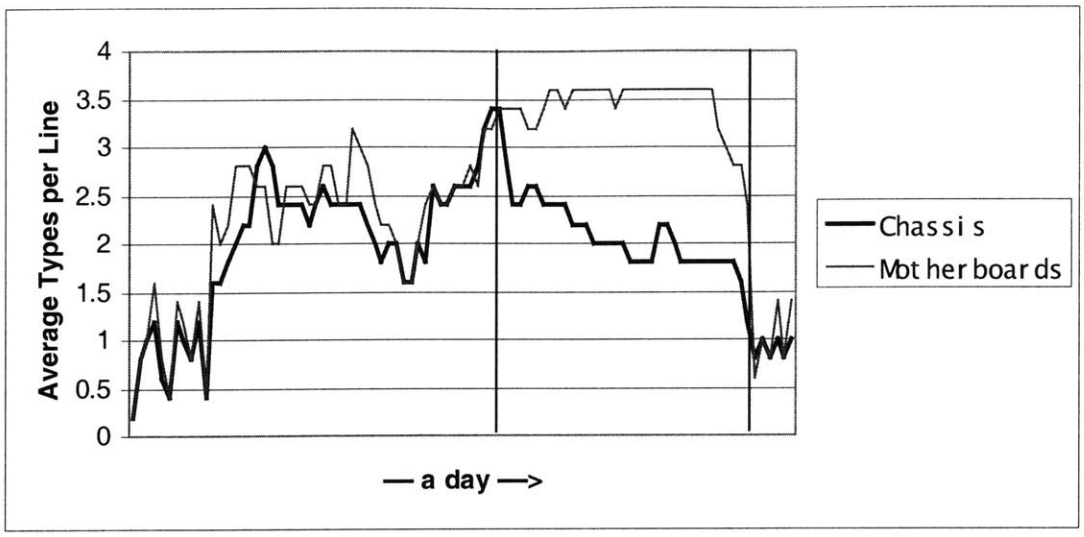


**Figure 5-2: Fluctuations of Component Types during a Day**

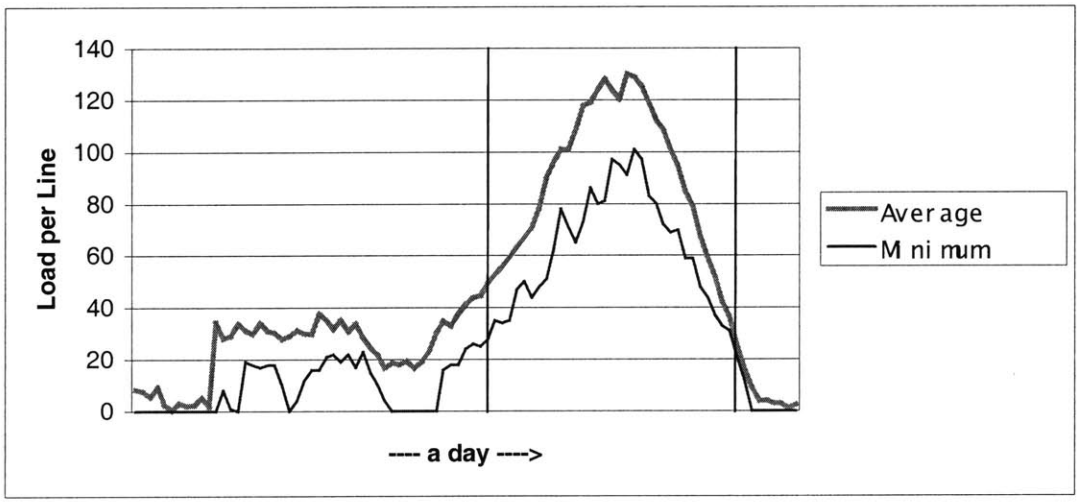


**Figure 5-3: Workloads during a Day**

These pictures demonstrate a common theme that is seen most of the simulated working days. This deterioration of the type mixture that is later resolved as a steady state is reached. Figure 5-4 shows an example that when the threshold is lower, the peak of the average number of types per line is earlier.

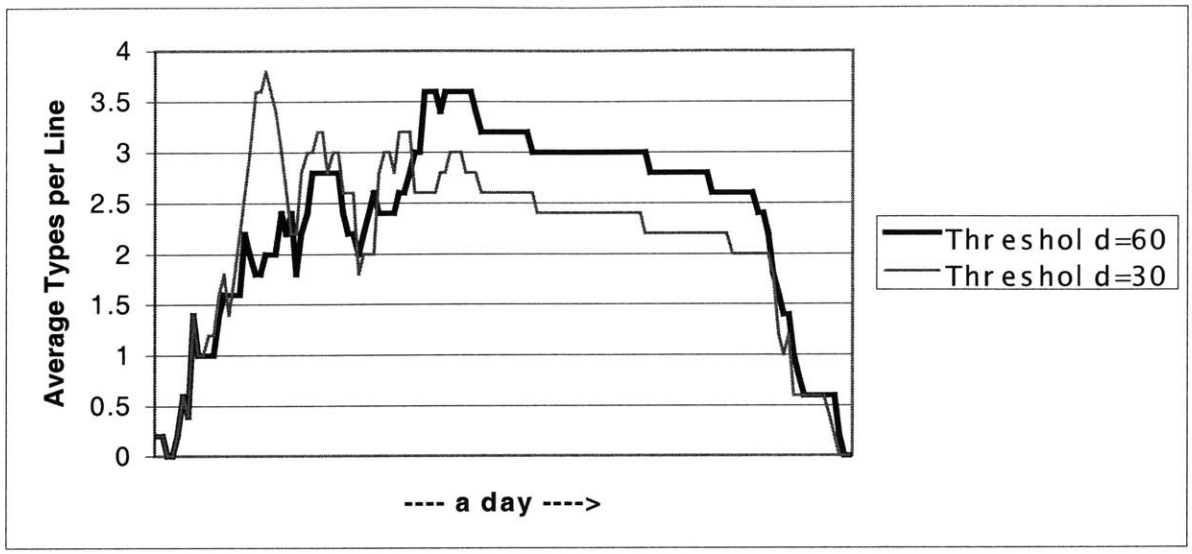


**Figure 5-4: Component Types in respect to Threshold**

The algorithm resolves the downtime problem by feeding starving lines first. Consequently, the distribution of the chassis and motherboards on the lines may deteriorate. Pathological cases where extreme distribution of ordered configurations might easily be drawn up. This weakness is a function of the algorithm's online heuristic character. Unless we use an "oracle", that can foresee future orders, we prefer having the lines work at the expense of the occasional accumulation of different types on the lines. However, looking at historical data, these events occur infrequently.

## *5.2 Analysis of Improvement Phase*

The improvement phase attempts to improve the component type fits and the workload balance. For each order, we check if it may be moved to some other line in a way that improves the fit in terms of chassis and motherboards and, at the same time, does not cause greater imbalance. In order to compare two lines in terms of fit, we can express the ranking in a formula:

$$W_1 \cdot \textit{aggregation function of} \text{ (Max Class of Chassis } \textit{and} \text{ Max Class of Motherboards)}$$

$$+ W_2 \cdot \textit{aggregation function of} \text{ (Target Class of Chassis } \textit{and} \text{ Target Class of Motherboards)}$$

$$+ W_3 \cdot \text{percentage metric}$$

where $W_i$'s are of different orders of magnitude such that the sequential optimization is preserved.

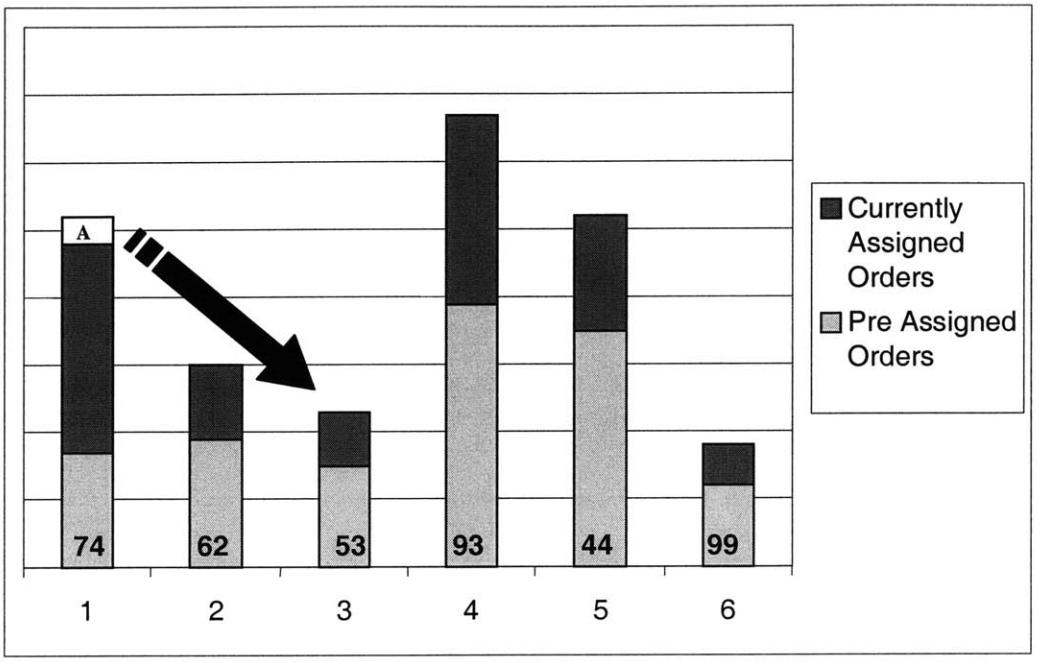


**Figure 5-5: Improvement Phase**

In figure 5-5, there are six lines with their respective rank values in respect to order A on line 1. Line 5 has the lowest ("best") rank value of 44, then 3, 2, 1, 4 and 6. Transferring order A to line 5 is unacceptable due to the fact that this will cause a greater imbalance than is already present in the system. The next candidate is therefore line 3. This transfer is acceptable since line 3 is less loaded than line 1.

Any order can only be moved to a line that has a smaller workload. Thus, the first order examined, from the most heavily loaded line can move to any of the other lines. An order from the third heaviest line can move to all lines but the top three. An order that has moved will usually not move again during the improvement phase. The only situation in which an order may move more than once is when the only difference between its current assigned line and another line is the percentage component of the rank formula. This component may get better during the

improvement phase iterations. Otherwise, a ranking of a line (in relation to a specific order) can never improve as more orders are added to the line (the classes may only "deteriorate"). The iterations cease once all orders cannot be moved. For simplicity we do not allow more than one optimization, since we may execute numerous iterations for minimal improvement. In summary, most orders will not be moved more than once and the process is finite as each move strictly improves the overall picture.

Simulations have shown that the improvement phase has very little effect. These simulations emphasize in which cases an "improvement" can actually be made. Since the algorithm is greedy by design, each order is assigned to the line best fitting its components (ignore starvation issues). Thus, an improvement will mean that a component makeup of a different line is suddenly more favorable in respect to the order in question. Since accumulation of orders cannot improve the class situation (it can worsen it by introducing new types), the only possible favorable change can be an improvement of the last, percentage based, component of the ranking. Apparently, such a change rarely occurs.

The improvement phase does have an effect on orders assigned when some lines are outside of the eligible window. This situation often happens around the starvation threshold. If a subset of the lines is below this threshold then new orders will be assigned to these lines only. Assuming all or most lines during a single batch cross the threshold, the improvement phase may reassign some of the orders that were assigned under threat of starvation.

Two scenarios are given as an example. The same batch (based on historic data) is assigned to five empty lines (all with no maximum limit and a target of one). In the first scenario, the starvation threshold is 30 while in the second there is no starvation threshold. The orders are in the form of *[chassis; motherboard; quantity]*. Each order is assigned to one of the lines 0 to 4.

Scenario 1:

[M;B;1]→0 [L;B;1]→2 [L;JA;7]→4 [L;JA;1]→4 [L;JA;7]→4[M;JA;15]→1
[S;JA;8]→3 [S;JA;4]→3 [T;JA;21]→4[T;JA;35]→1[T;JA;18]→3[L;J;1]→2
[L;J;6]→2 [M;J;5]→0 [M;J;5]→0 [S;J;7]→3 [S;J;1]→0 [T;J;47]→2
[L;S;1]→0 [M;S;4]→0 [S;S;3]→0 [T;S;2]→0 [T;S;14]→0 [S;T;4]→3
[L;T;5]→4 [T;T;2]→4 [T;T;6]→3

The makeup of the lines after the assignment:

| | | | | |
|---|---|---|---|---|
| LINE 0: | Total: | 36 | Ch: | S=4 T=16 L=1 M=15 |
| | | | Mb: | J=11 B=1 S=24 |
| LINE 1: | Total: | 50 | Ch: | T=35 M=15 |
| | | | Mb: | JA=50 |
| LINE 2: | Total: | 55 | Ch: | T=47 L=8 |
| | | | Mb: | J=54 B=1 |
| LINE 3: | Total: | 47 | Ch: | S=23 T=24 |
| | | | Mb: | J=7 JA=30 T=10 |
| LINE 4: | Total: | 43 | Ch: | T=23 L=20 |
| | | | Mb: | JA=36 T=7 |

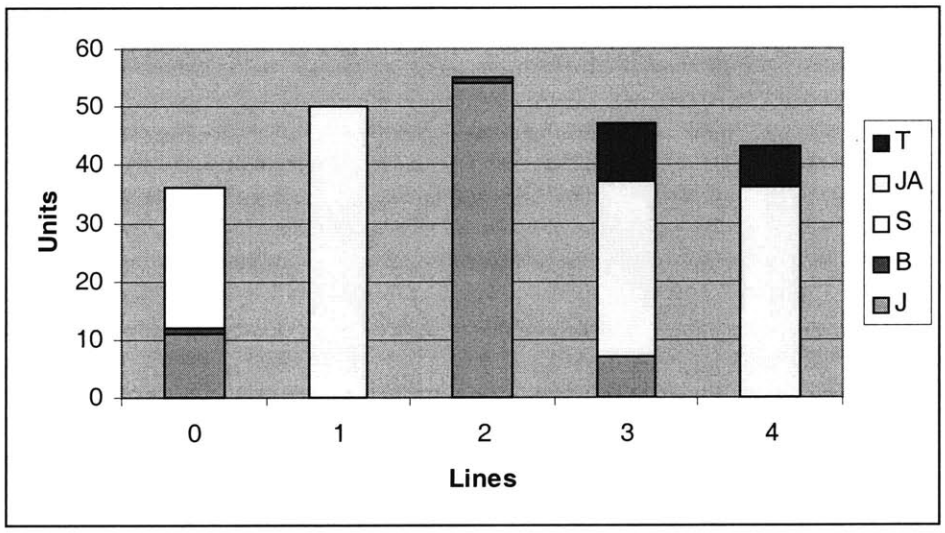


**Figure 5-6: Distribution of Work Before Improvement (Motherboards Displayed)**

The following orders are switched during the improvement phase:

```
[L;B;1] switched from 2 to 0
[S;J;7] switched from 3 to 0
[T;T;2] switched from 4 to 3
```

The makeup of the lines after the improvement:

```
LINE 0:    Total:    44        Ch:   S=11  T=16  L=2   M=15
                               Mb:   J=18  B=2   S=24
LINE 1:    Total:    50        Ch:   T=35  M=15
                               Mb:   JA=50
LINE 2:    Total:    54        Ch:   T=47  L=7
                               Mb:   J=54
LINE 3:    Total:    42        Ch:   S=16  T=26
                               Mb:   JA=30 T=12
LINE 4:    Total:    41        Ch:   T=21  L=20
                               Mb:   JA=36 T=5
```

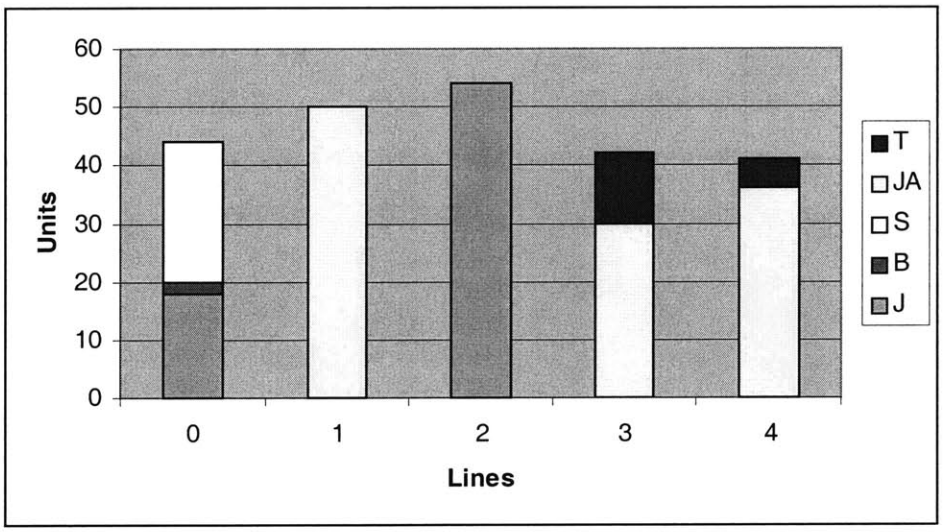


**Figure 5-7: Distribution of Work after Improvement (Motherboards Displayed).**

The three orders that were reassigned were initially assigned under the threat of starvation. Before the improvement the average number of chassis per line is 2.4 and the average number of motherboards is 2.2. The minimum load is 36 and the maximum is 55. After the improvement the average number of chassis per line remains 2.4 but the average number of

motherboards decreases to 1.8. The minimum load is 41 and the maximum is 54, indeed more balanced.

Scenario 2:

```
[M;B;1]→0  [L;B;1]→2  [L;JA;7]→4 [L;JA;1]→4 [L;JA;7]→4[M;JA;15]→1
[S;JA;8]→3 [S;JA;4]→3 [T;JA;21]→4[T;JA;35]→4[T;JA;18]→4[L;J;1]→2
[L;J;6]→2  [M;J;5]→1  [M;J;5]→1  [S;J;7]→3  [S;J;1]→3  [T;J;47]→4
[L;S;1]→2  [M;S;4]→0  [S;S;3]→0  [T;S;2]→2  [T;S;14]→2 [S;T;4]→3
[L;T;5]→3  [T;T;2]→4  [T;T;6]→4
```

The makeup of the lines after the assignment:

```
LINE 0:    Total:    8          Ch:   S=3   M=5
                                 Mb:   B=1   S=7
LINE 1:    Total:    25         Ch:   M=25
                                 Mb:   J=10  JA=15
LINE 2:    Total:    25         Ch:   T=16  L=9
                                 Mb:   J=7   B=1   S=17
LINE 3:    Total:    29         Ch:   S=24  L=5
                                 Mb:   J=8   JA=12 T=9
LINE 4:    Total:    144        Ch:   T=129 L=15
                                 Mb:   J=47  JA=89 T=8
```

Notice the conspicuous load imbalance.

The following order is switched during the improvement phase:

```
[T;J;47] switched from 4 to 1
```

The makeup of the lines after the improvement:

```
LINE 0:    Total:    8          Ch:   S=3   M=5
                                 Mb:   B=1   S=7
LINE 1:    Total:    72         Ch:   T=47  M=25
                                 Mb:   J=57  JA=15
LINE 2:    Total:    25         Ch:   T=16  L=9
                                 Mb:   J=7   B=1   S=17
```

```
LINE 3:    Total:     29        Ch:   S=24  L=5
                                Mb:   J=8   JA=12 T=9
LINE 4:    Total:     97        Ch:   T=82  L=15
                                Mb:   JA=89 T=8
```

Since no orders were assigned under the threat of starvation less improvement is done during the improvement phase. The order that is improved is one that is moved due to accumulation of motherboards on line 4. Before the improvement the average number of chassis per line is 1.8 (compare to 2.4) and the average number of motherboards is 2.6. The minimum load is 8 (compare to 36) and the maximum is 144 (compare to 55). After the improvement the average number of chassis per line increases to 2.6 and the average number of motherboards decreases to 2.4. The minimum load is still 8 but the maximum is 97.

## 5.3 Analysis of Different Queueing Disciplines

Queue disciplines specify the disposition of blocked customers, those customers who find all servers busy. Hence, they specify whether customers stay in the system, their priority over other waiting customers or customers being served (known as preemptive), etc. Common disciplines include First Come First Serve, Last Come First Serve, priority queues – where each customer has a relative priority and more. It is interesting to see how various disciplines may affect our heuristic. Since we accumulate a batch of orders we can decide to sequence the batch in a predefined order. We have investigated five approaches:

1. Unsorted – as they arrived from sales.
2. Sorted first by chassis then by motherboard.
3. Sorted first by motherboard then by chassis.
4. Sorted by quantity ascending.
5. Sorted by quantity descending.

Below are the results pertaining to the first scenario from the previous section. All after construction and improvement phases.

**Sorted by chassis and then by motherboard.**

```
LINE 0: Total:  40
        Chassis:        T = 23  L = 6   M = 11
        Motherboards:   J = 10  B = 2   JA = 21 T = 7
LINE 1: Total:  45
        Chassis:        S = 7   L = 1   T = 37
        Motherboards:   JA = 35 S = 6   T = 4
LINE 2: Total:  45
        Chassis:        S = 20  T = 18  L = 7
        Motherboards:   J = 15  JA = 30
LINE 3: Total:  47
        Chassis:        T = 47
        Motherboards:   J = 47
LINE 4: Total:  48
        Chassis:        T = 14  L = 15  M = 19
        Motherboards:   S = 18  JA = 30
```

**Sorted by motherboard and then by chassis.**

```
LINE 0: Total:  36
        Chassis:        S = 7   T = 18  L = 7   M = 4
        Motherboards:   B = 1   S = 24  T = 11
LINE 1: Total:  57
        Chassis:        T = 47  M = 10
        Motherboards:   J = 57
LINE 2: Total:  34
        Chassis:        T = 18  M = 16
        Motherboards:   B = 1   JA = 33
LINE 3: Total:  55
        Chassis:        S = 20  T = 35
        Motherboards:   J = 8   JA = 47
```

```
LINE 4: Total:  43
        Chassis:        T = 21  L = 22
        Motherboards:   J = 7   JA = 36
```

**Sorted by quantity - ascending**

```
LINE 0: Total:  56
        Chassis:        T = 56
        Motherboards:   JA = 56
LINE 1: Total:  29
        Chassis:        S = 11  T = 4   M = 14
        Motherboards:   J = 22  S = 7
LINE 2: Total:  34
        Chassis:        T = 18  M = 16
        Motherboards:   B = 1   JA = 33
LINE 3: Total:  32
        Chassis:        S = 16  L = 16
        Motherboards:   B = 1   JA = 27 T = 4
LINE 4: Total:  37
        Chassis:        T = 24  L = 13
        Motherboards:   J = 7   S = 17  T = 13
```

**Sorted by quantity descending**

```
LINE 0: Total:  49
        Chassis:        L = 1   T = 47  M = 1
        Motherboards:   B = 2   J = 47
LINE 1: Total:  44
        Chassis:        S = 24  M = 20
        Motherboards:   J = 13  JA = 27 T = 4
LINE 2: Total:  44
        Chassis:        T = 35  M = 9
        Motherboards:   J = 5   S = 4   JA = 35
LINE 3: Total:  49
        Chassis:        L = 27  T = 22
        Motherboards:   J = 7   JA = 15 S = 14  T = 13
```

```
LINE 4: Total:  44
        Chassis:          S = 3    T = 41
        Motherboards:     S = 5    JA = 39
```

The results are shown in the table 5-2:

| Discipline | Average Chassis | Average Motherboards | Weighted Average Formula | Minimal Load | Maximal Load | Delta |
|---|---|---|---|---|---|---|
| **Unsorted** | **2.4** | **1.8** | **3.66** | 36 | 55 | 19 |
| Chassis then MB | 2.6 | 2.4 | 4.28 | 40 | 48 | 8 |
| MB then Chassis | 2.4 | 2 | 3.8 | 36 | 57 | 21 |
| **Quantity Ascending** | **2** | **2.2** | **3.54** | 29 | 56 | 27 |
| Quantity Descending | 2.2 | 2.8 | 4.16 | 44 | 49 | 5 |

**Table 5-2: Simulation Results for Various Disciplines**

In this case, unsorted dominates both "same component type first" disciplines and quantity ascending dominates quantity descending in respect to average chassis and motherboards on a line. There is also a high correlation between the average component type per line and the gap, or balance on the lines. In other words, the better the component type the worse off the balance, as expected. Purely based on the weighted average formula, the formula we used earlier to aggregate the metrics for chassis and motherboards ($c + 0.7m$), it seems that the quantity descending discipline has a slight edge over unsorted, although the imbalance is greater. It is further interesting to see that only the unsorted and quantity ascending disciplines have no line that stages more than three types of either component. On the other hand, since this a scenario that occurs before a steady state is reached its results may be misleading.

A simulation similar to the one run earlier, which is based on 5911 data points (batches of orders), sheds some insight. Using the same parameters as before, this time with various disciplines reveals that sorting by chassis then motherboards ("Same Chassis First") is better over the long run. By observing the weighted average metric we are able to conclude that this

discipline is slightly superior to others. The table below shows in how many of the data points the discipline in the horizontal axis was better than the discipline in the vertical axis. For example, "Quantity Descending" was better than "Quantity Ascending" 1474 out of 5911 times.

| | Unsorted | Quantity Ascending | Quantity Descending | Chassis then Motherboard | Motherboard Then Chassis |
|---|---|---|---|---|---|
| **Unsorted** | | 1532 | 1525 | **1827** | 1424 |
| **Quantity Ascending** | 1369 | | 1474 | **1538** | 1419 |
| **Quantity Descending** | 1376 | 1413 | | **1621** | 1495 |
| **Chassis then Motherboard** | 1118 | 1347 | 1357 | | 1265 |
| **Motherboard Then Chassis** | 1367 | 1498 | 1415 | **1653** | |

**Table 5-3: Comparison of Weighted Average Formula**

According to table 5-3, the discipline of "Same Chassis First" dominates in an empiric environment, in some cases significantly, all other disciplines. It is further interesting to see that the unsorted discipline is actually dominated by all others.

In order to minimize total production time per unit, we should optimally use a "Shortest Order First" discipline. If all orders have the same assembly time then the correct theoretical policy is to assign each single order to the shortest current queue (assuming away the complications of the real system).

# 6 Chapter Six Conclusion

## *6.1 Review*

This paper described the line assignment problem with which Dell Computer Corporation has been grappling. Dell Computer, which considers its logistics and assembly operations as a core competence, is pursuing further improvement in these areas. In addition, many of Dell strategic focuses tie in directly with this problem – the build to order approach and just in time inventories also provide key advantages over the competition.

The issue of line balancing became of greater concern with the implementation of the new supply chain management and demand fulfillment software systems. Previously, a person would "pull" available to build orders onto her line. Thus, she and her colleagues on the neighboring lines had a limited ability to manually maintain a balanced distribution of workload. The new demand fulfillment was brought on board in part to further reduce inventories and material handling as well as create a controlled schedule sequence. Consequently, the decision on assignment of orders to lines had to be resolved efficiently and systematically. The new demand fulfillment software did not address the assignment of orders to lines. The system expected to receive as input the destination assembly line of each order and then optimized based on other criteria such as due date, business priority and capacity.

The line assignment problem encompasses multiple objectives. These objectives are to maintain a balanced distribution of work among the assembly lines and to preserve a low mix of system configurations on each line. As a result, downtime is reduced while maintaining an efficient process.

Several constraints limit the optimization of these objectives. These constraints range from physical material restrictions through special process requirements for premium orders to the time the algorithm consumes. Furthermore, the solution must be simple enough in order to be accepted and understood by the various stakeholders at the company.

Our first approach towards creating a solution for this problem involved Integer Programming optimization. The main obstacle encountered with this approach is its unreliability in respect to computation time. In addition, as the problem was articulated more precisely, the number of variables increased and the solution space expanded dramatically.

Due to these obstacles we investigated a more computationally efficient heuristic approach. The motivation is that our real goal is to have a good, acceptable solution rather than an absolute "optimal" solution, whatever "optimal" may be. The intricacy of our problem is due to the three independent objectives we wish to optimize simultaneously. These objectives typically do not coincide. Choosing one line over another will ordinarily improve our position in respect to one objective yet compromise the other.

The general structure of the proposed solution involves two phases: a construction phase and an improvement phase. The input to the solution is a batch of orders that have accumulated. During the construction phase we will iterate through the one at a time and assign each order to a line. The construction phase consists of three main steps. During each step the set of eligible candidates, initially the entire set of lines, decreases in size and eventually we end up with one alternative. The first step eliminates visibly ineligible candidate lines – lines that are not an option due to hard and binding constraints. The second step selects only lines that are within a certain time window – lines that are the least loaded. The third step selects the line that has the best fit in terms of the component types. During the following phase, the improvement phase, the algorithm tries to improve the assignments by trying to move orders to better fitting lines. We later introduced additional enhancements to this solution such as generalization of targets.

## 6.2 Issues for Further Research

- Batch size

Different batch sizes may yield different implications for the assignment of orders to lines. There are clear advantages for either longer accumulations of orders or shorter ones. The shorter the batches, a single order at the extreme, the more real time view the Planner has of the system. Dell wants to maintain a near real time picture, which will enable maximal flexibility in planning. In addition, this approach does not rely on the existence of any backlog. On the other hand, a larger accumulation of orders may yield better planning. At an extreme, if the orders were all known before planning an optimal schedule could be created in theory. However, it would also mean reliance on large backlogs, inability to maintain near real time views of the system and relatively heavier tasks computationally.

Accurate measurement of the implications of batch size on scheduling requires more extensive integration with the Planner software. It is clear that a long time horizon may yield better assignments. However, only control of the actual sequence may allow exploitation of this issue. Since our scope does not include the ability to sequence, the result may be that more component types will accumulate in every line – contrary to our objective.

- Upstream Integration

Tighter integration with the Sales department may yield better assignment decisions. Sales, the source of all orders, may possess more information about future orders than is visible at Operations. Due to the build to order nature of Dell's operations, planning for a long horizon is impractical. In order to mitigate this difficulty, mechanism should be put in place to access all information that is available anywhere within Dell. Increased visibility to upstream orders may enable an assignment algorithm to make better decisions and prevent locally optimal decision that may result in later penalties.

- Enhanced Modeling

More accurate modeling by the Planner software can yield a more optimal schedule. Assuming a stronger integration between the Planner software and the assignment algorithm, modeling of phenomena such as the minor assembly rate increase for similar systems will be possible. This phenomenon is referred to in Chapter 3 – faster assembly rate for similar consecutive systems. This modeling will provide a more precise representation of real assembly operations. Currently, no interdependence between systems on the production lines exist in the model, hence it is viewed as workload conserving system. However, in reality there are such interactions that should be explored in greater detail.

- Enhanced Planner

The assignment solution should be incorporated with the scheduling logic within the Planner. It is unreasonable that the assignment of orders to lines is treated and viewed as an independent function. Rather, the decision on which line a system should be assembled should be postponed to the last instance when the state of the lines at that exact moment is known, as well as the maximal information about the orders waiting in queue. Results for these scheduling problems may be applied.

## Bibliography

Cooper, Robert B.: *Introduction to Queueing Theory,* New York: The Macmillan Company, 1972.

Fonesca, Carlos M., Fleming, Peter J.: *An Overview of Evolutionary Algorithms in Multiobjective Optimization,* Evolutionary Computation, 3(1): 1-16, spring 1995.

Hwang, Ching Lai: *Multiple Objective Decision Making - Methods and Applications,* Springer-Verlag, Berlin, 1979.

Keeney, Ralph L. and Raiffa Howard: *Decisions with Multiple Objectives,* New York: John Wiley & Sons, 1976.

McWilliams, Gary: *Dell Looks for Ways to Rekindle the Fire It Had as an Upstart,* Wall Street Journal August 31, 2000.

Newell, G. F.: *Applications of Queueing Theory,* London: Chapman and Hall Ltd., 1971.

Strickland, A. J. and Thompson Arthur A.: *Strategic Management,* Eleventh Edition, Homewood, IL: McGraw Hill Companies, 1999.